НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Метод та програмні засоби прискореного експоненціювання на полях Галуа»

Виконала:
студентка IV курсу, групи ІО-61
Ольга КОТ _____

Керівник:
Доц. каф. ОТ, к. т. н.
Олександр МАРКОВСЬКИЙ _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
Валерій СІМОНЕНКО _____

Рецензент:
Декан ФПМ, доктор технічних наук, професор
Іван ДИЧКА

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський) Спеціальність – 123 «Комп'ютерна інженерія» Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проект студентці
Ользі КОТ

1. Тема проекту «Метод та програмні засоби прискореного експоненціювання на полях Галуа», керівник проекту Марковський Олександр Петрович, доцент, к. т. н., затверджені наказом по університету від «07» травня 2020 р. №1081-с

2. Термін подання студентом проекту 02.06.2020

3. Вихідні дані до проекту технічна документація

4. Зміст пояснювальної записки

Аналіз використання алгебри полів Галуа в сучасних алгоритмах захисту інформації метод прискорення експоненціювання на полях Галуа за рахунок технології Монтгомері
Розробка програмних засобів для реалізації методу обчислення експоненти на полях Галу з використанням редукції монтгомері.

5. Перелік графічного матеріалу

Схема алгоритму класу, функціональна схема, принципова схема.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Валерій СІМОНЕНКО		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Затвердження теми роботи*		
2.	*Вивчення та аналіз завдання*		
3.			
4.			
5.			
6.	*Оформлення пояснювальної записки*	13.04.2020-25.05.2020	
7.	*Передзахист*	24.05.2020	
8.	*Захист*		

Студент Ольга КОТ

Керівник Олександр МАРКОВСЬКИЙ

Анотація

Бакалаврський дипломний проект присвячений проблемі прискорення експоненціювання на полях Галуа, операції що лежить в основі широкого кола сучасних криптографічних алгоритмів.

Бакалаврській роботі досліджено можливості використання рекурсії Монтгомері для прискорення мультиплікативних операцій на полях Галуа. На основі отриманих теоретичних результатів, розроблено метод прискореного експоненціювання на полях Галуа, з використанням рекурсії Мотгомері. Наведено детальний опис експоненціювання на полях Галуа з використанням рекурсії Монтгомері, а також результати, що доводять ефективність прискорення. Запропонований метод ілюстровано числовими прикладами. Доведено, що розроблений метод забезпечує значне прискорення в порівнянні з класичною процедурою експоненціювання на полях Галуа, за рахунок заміни операції поліноміального ділення операцією зсуву.

Annotation

Bachelor's project is deals with the problem of accelerating of Galois fields exponentiation which is based operation of a wide range of modern data security algorithms.

In bachelor's project the possibility of using Montgomery recursion for accelerating of multiplication operation on Galois fields is investigated. Based on the obtained theoretical results, a new method of accelerated Galois fields exponentiation has been developed by using Montgomery recursion. The article consists a detailed description of Galois fields exponentiation with using Montgomery recursion, its theoretical justification, and results of the evaluation of the proposed method effectiveness. The proposed method is illustrated by numerical examples. It is proven, that the proposed method provides significant

acceleration, compared to classical processing of Galois fields exponentiation by replacing the polynomial division operation by a shift operation.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 4682.01.000 ВП	Відомість дипломного проекту	1	
3	А4	ДП 4682.02.000 ТЗ	Технічне завдання	4	
4	А4	ДП 4682. 03.000 ПЗ	Пояснювальна записка	68	
5	А4	ДП 4682. 04.000 Д1	Схема	1	
6	А3	ДП 4682. 05.000 Д2	Схема	1	
7	А3	ДП 4682. 06.000 Д3	Схема	1	
8	А4	ДП 4682. 07.000 Д4	Лістинг реалізації ідентифікації	5	

Зм.	Арк.	№ докум.	Підпис	Дата	ДП 4682.01.000 ВП		
Розробивла		Кот О. С.			Відомість дипломного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Марковський О.П.					
Н. контр.		Сімоненко В. П.				НТУУ "КПІ", ФІОТ, каф. От, гр. ІО-61	
Затверд.							

Технічне завдання до дипломного проекту

ЗМІСТ

					ДП 4682.02.000 ТЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробивла		Кот О. С.			Технічне завдання	Літ.	Аркуш 1 / Аркушів 4
Перевір.		Марковський О.П.					
						НТУУ "КПІ", ФІОТ, каф. Ом, гр. ІО-61	
Н. контр.		Сімоненко В. П.					
Затверд.							

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічне завдання поширюється на розробку нового методу прискорення експоненціювання на полях Галуа та програмні засоби його реалізації

Область застосування – системи криптографічного захисту інформації в комп'ютерних комплексах і мережах.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня "бакалавр комп'ютерної інженерії", затверджене кафедрою спеціалізованих комп'ютерних систем Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Мета проекту полягає в підвищенні ефективності засобів криптографічного захисту інформації в основі яких лежить операція експоненціювання на кінцевих полях Галуа за рахунок прискорення цієї базової для широкого кола алгоритмів криптографічного захисту операції шляхом застосування для редукції технології Монтгомері.

Розробка призначена для прискорення реалізації криптографічних протоколів на основі еліптичних кривих, а також модифікованих протоколів автентифікації віддалених користувачів та цифрового підпису електронних документів.

4. ДЖЕРЕЛА РОЗРОБКИ

4.1. Кот О.С. Організація прискореного експоненціювання на полях Галуа з використанням редукції Монтгомері / О.С.Кот, О.П. Марковський // Альманах науки.- 2020.- № 3 (36).- С.34-37.

4.2. Markovskyi Oleksandr The Employment of Montgomery reduction for acceleration of exponent on Galoise fields calculation / O. Markovskyi, V. Masimyk, O.Kot // Proceeding of International Conference on Security, Fault Tolerance, Intelligence" (ICSFTI2020), 13-14 травня 2020, м. Київ.- P.44-49.

4.3. Марковський О.П. Використання алгебри полів Галуа для реалізації концепції «нульових знань» при ідентифікації та автентифікації віддалених / О.П.Марковський, Захаріудакіс Ліфтеріс, Максимук В.Р. // Електронне моделювання. Збірник наукових праць. - 2017.- Т.6, №39. - С.33-45.

4.4. Марковський О.П. Технологія цифрового підпису DSA на основі арифметики полів Галуа / О.П. Марковський, Саїдреза Махмали, Ісаченко Г.В. // Вісник націо¬нального технічного університету України "КПІ". Інформатика, управління та обчислювальна техніка. — 2012. - № 55. - С. 34 — 41.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

5.1.1 Для прискорення експоненціювання на кінцевих полях Галуа використовувати зменшення витрат часу на редукцію проміжних результатів шляхом адаптації технології Монтгомері до особливостй алгебри кінцевих полів Галуа.

5.1.2 Для практичної реалізації використання технології Монтгомері на полях Галуа окремо розробити модифіковані алгоритми редукції кодів з використанням технології Монтгомері на кінцевих полях, алгоритм множення на полях Галуа з застосуванням редукції Монтгомері; алгоритму експоненціювання на полях Галуа з використанням модифікованого множення та редукції проміжних результатів експоненціювання за технологією Монтгомері.

5.1.3 Ступінь утворюючого поліному кінцевого поля Галуа - не менше 2048.

5.1.4 Для корекції отриманого результату використовувати код, розрядністю рівною ступеню утворюючого поліному, що містить одиницю в старшому розряді і нулі – в усіх інших та мультиплікативну інверсію цього коду.

5.1.5 Порівняння швидкодії виконати по відношенню з класичним алгоритмом експоненціювання на полях Галуа зі старших розрядів коду експоненти.

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 10
- Visual Studio 2017
- C++11

5.3. Вимоги до апаратного забезпечення

- Процесор рівня Intel i5 і вище.
- Оперативна пам’ять не менше 500 МБ.
- Вільне місце на жорсткому диску не менше 100 МБ.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	20.12.2019
Створення та узгодження технічного завдання	15.01.2020
Вивчення літературних джерел	27.01.2020
Розробка алгоритмів редукції, множення на полях Галуа та експоненціювання на полях Галуа	15.04.2020
Розробка програмної моделі	01.05.2020
Відлагодження програми та виправлення помилок	15.05.2020
Оформлення документації дипломного проекту	06.06.2020

Пояснювальна записка
до дипломного проекту
на тему: «Метод та програмні засоби прискореного експоненціювання на полях Галуа»

Київ – 2020 року

ЗМІСТ

ВСТУП

З початком нового тисячоліття інформаційна інтеграція на основі комп'ютерної обробки даних та мережевих технологій стає одним із найвагоміших факторів технологічного прогресу людства. Швидкий прогрес поглиблення інтеграційної інтеграції стимулює динамічний розвиток систем регулювання доступу та захисту даних в інтегрованому інформаційному середовищі а також технологій передачі цифрових даних. Ефективне існування інтегрованого інформаційного середовища не можливе без цілісної системи регулювання доступу до даних, контролю автентичності та цілісності, їх забезпечення авторських прав та конфіденційності. В рамках такої системи до теперішнього часу створена сукупність протоколів захисту інформації. Їх більша частина базується на криптографічних механізмах з відкритим ключем, математичною основою якої є мультиплікативні операції модулярної арифметики з числами, довжина яких суттєво перевищує розрядність процесорів.

Реалією сучасного етапу розвитку технологій захисту інформації є зростання значимості швидкості реалізації функцій захисту інформації. Це зумовлено наступними чинниками:

- з розширенням інформаційної інтеграції комп'ютерні системи, з точки зору обміну даними, все більше стають системами колективного доступу ефективне функціонування яких може бути забезпечено лише за умови високої продуктивності засобів захисту процесів інформаційної взаємодії;

- можливість інтегрування ресурсів, що можуть бути потенційно використані для порушення захисту диктує постійне зростання розрядності чисел, що використовуються в криптографічних механізмах, що має наслідком експоненційне зростання обчислювальних ресурсів для їх реалізації;

- зростання використання в якості термінальних компонентів інтегрованого інформаційного середовища малопотужних мобільних обчислювальних пристроїв, що мають підтримувати протоколи захисту даних.

Зумовлена вказаними чинниками потреба в підвищенні продуктивності реалізації функцій захисту інформації і, в першу чергу, алгоритмів з відкритим ключем, не може бути вирішена за рахунок зростання швидкодії комп'ютерних систем і вимагає розробки спеціальних підходів, методів. Одним із найбільш перспективних шляхів прискорення алгоритмів несиметричної криптографії є використання перетворень на кінцевих полях Галуа.

Базовою обчислювальною операцією більшості сучасних алгоритмів несиметричної криптографії, таких, зокрема, як Ель-Гамаля, RSA, Schoor, DSA є модулярне експонеціювання над числами, розрядність яких значно перевищує розрядність процесора. Ця операція має явно виражений послідовний характер і тому не може бути розпаралелена. Використання багаторозрядних операндів при використанні традиційної арифметики зумовлює значні затримки виконання всіх мікрооперацій по причині помітного часу на формування переносів з 1-го до 2048-го розряду. Крім того, операції віднаходження залишку від ділення чисел такої розрядності потребують значного часу, оскільки в сучасних мікропроцесорах відсутні відповідні команди.

З точки зору обчислювальної реалізації використання в якості базової операції експоненціювання на полях Галуа надає суттєві переваги, які дозволяють прискорити процес експоненціювання:

- в арифметиці кінцевих полів кожен розряд 2048-чисел оброблюється незалежно: це дозволяє виключити обробку переносів і таким чином суттєвим чином прискорити обчислення;

- операція редукції на полях Галуа може виконуватися значно швидше.

Особливо чітко переваги використання арифметики полів Галуа в порівнянні з традиційною арифметикою з точки зору швидкості обчислень проявляються при апаратній реалізації.

Виникає задача ретельного дослідження обчислювальної процедури експоненціювання на полях Галуа в ракурсі пошуку нових можливостей подальшого прискорення обчислення.

Таким чином, тема бакалаврської роботи, направленої на створення методів та засобів підвищення продуктивності реалізації функцій захисту інформації в комп'ютерних системах за рахунок використання в якості математичної основи криптографічного кодування спеціальної алгеб¬ри кінцевих полів Галуа, яка дозволяє спростити і прискорити обчислення є актуальною для сучасного етапу розвитку комп'ютерних технологій.

РОЗДІЛ 1

АНАЛІЗ ВИКОРИСТАННЯ АЛГЕБРИ ПОЛІВ ГАЛУА В СУЧАСНИХ АЛГОРИТМАХ ЗАХИСТУ ІНФОРМАЦІЇ

1.1 Критерії ефективності алгоритмів криптографічного захисту

Основними завданнями для систем захисту інформації [1] в сучасних обчислювальних системах та мережах є:

- Забезпечення захисту інформаційних повідомлень користувачів, від можливого доступу до неї третіх осіб (несанкціоноване читання файлів), що передаються через комп'ютерні мережі, а також зберігаються як файли на магнітних чи оптичних носіях.

- Забезпечення захисту від можливого втручання в файлі чи повідомлення третіми особами, внесення в них змін, а саме забезпечення автентичності та цілісності файлів та інформаційних повідомлень, що зберігаються в пам'яті і були передані по мережам.

- Забезпечення автентифікації та ідентифікації користувачів, для захисту інтегрованих систем обробки інформації та баз даних, від можливого втручання та несанкціонованого доступу до них третіми особами.

Для того, аби вирішити ці задачі, необхідно застосувати складний комплекс щільно пов'язаних між собою засобів організації (протоколів), алгоритмів, програмних та технічних засобів [3]. Криптографічні алгоритми також є важливим елементом цих комплексів. Вони використовуються для вирішення усіх завдань забезпечення відповідного рівня інформаційної безпеки в комп'ютерних системах і мережах, які були зазначені вище. Криптографічні алгоритми, на відміну від інших технічних засобів, мають високу гнучкість та можливість доволі швидкої зміни захисних властивостей, шляхом зміни ключа. Також криптографічні алгоритми майже

не залежать від розвитку технічних засобів зламу, тому вони якнайкраще підходять для вирішення усіх віще зазначених питань.

При аналізі та оцінці криптографічних алгоритмів перш за все важливим є визначення основних критеріїв їх якості. Критерії залежать від протоколу, а також від конкретних умов застосування цього алгоритму. Окрім цього, для різних класів криптографічних алгоритмів об'єктивно існують свої власні критерії якості, несуттєві для алгоритмів інших класів. Так, наприклад, для алгоритмів "з відкритим ключем" найважливішим критерієм якості є час формування закритих і відкритих ключів. Проте цей самий критерій є не суттєвим, якщо розглядати криптографічні алгоритми інших класів. У загальному вигляді, для всіх криптографічних алгоритмів критерії якості можуть бути виділені таким чином:

- рівень захисту даних, який забезпечує криптографічний алгоритм. Мірою цього критерію у першому наближенні є витрати обчислювальних ресурсів, потрібні для злому створеного алгоритму захисту даних;

- швидкість криптографічної обробки інформації з використанням алгоритму на типових процесорних засобах;

- можливості ефективної організації обчислювального процесу для реалізації алгоритму з використанням спеціалізованих і проблемно-орієнтованих апаратних засобів;

- рівень стійкості даних, оброблених за допомогою криптографічного алгоритму, до впливу навмисних і ненавмисних перешкод у каналах передачі даних, а також при зберіганні закодованих інформаційних файлів на магнітних та оптичних носіях.

Проте необхідно зауважити суперечливий характер наведених критеріїв: поліпшення одного з них, зазвичай призведе до погіршення інших, так, що інтегральним показником стійкості криптографічного алгоритму необхідно вважати ступінь компромісної відповідності вище зазначених критеріїв, виходячи з конкретних умов і протоколу

використання алгоритму. Для визначення значимості окремо вибраних складових інтегрального критерію якості криптографічних алгоритмів, необхідно зважати на конкретні особливості використання криптографічного алгоритму.

Основою оцінки значущості складових критерію якості для комерційної криптографії є наступні основоположні принципи [2]:

1. Рівень стійкості криптографічного алгоритму інформаційного повідомлення чи файлів, повинен бути таким, щоб його розкриття було економічно невигідним, тобто прибуток від реалізації зламу(отримання доступу до інформації, що захищається алгоритмом) повинен бути завжди меншим, ніж вартість обчислювальних ресурсів, що необхідно затратити на злам криптографічного алгоритму.

2. Рівень витрат для реалізації захисного алгоритму повинен бути таким, щоб використання криптографічного алгоритму було економічно вигідним, тобто складність реалізації алгоритму, яку можна виміряти вартістю витрат ресурсів обчислювальних систем на реалізацію захисного алгоритму при передачі інформаційного повідомлення і супутніх втрат (наприклад, додаткова вартість через займання лінії зв'язку, якщо швидкість криптографічного перетворення менша, аніж швидкість передачі інформації по самій лінії) повинна бути менша ніж збитки, пов'язані з реалізацією несанкціонованого розкриття інформації, що захищається.

Зважаючи на вище наведені фундаментальні принципи організації криптографічного захисту інформації для недержавних, комерційних установ, можна зробити вивід, що в ідеалі має бути розроблена група добре досліджених і апробованих криптографічних алгоритмів для захисту інформації, що відрізняються рівнем крипостійкості і відповідно, швидкістю реалізації. На практиці таку політику активно проводять багато фірмам, зокрема, одним з представників є добре відома фірма виробник комп'ютерів – IBM, яка для вирішення окремих класів задач захисту

інформації створює сімейства алгоритмів, що відрізняються рівнем захищеності та витратами на практичну реалізацію [3].

В теорії [4], основою будь-якого криптографічного алгоритму є незворотне математичне перетворення, тобто таке перетворення, для якого визначено процедуру прямого перетворення, з якого неможливо отримати процедуру зворотного перетворення. В сучасних реаліях, в криптографії використовуються декілька типів таких незворотних математичних перетворень. Зокрема, в алгоритмах симетричного шифрування і хеш-алгоритмах використовуються незворотні перетворення булевої алгебри. Таким чином, можна визначити систему ортогональних булевих функцій, які трансформують вхідний вектор у вихідний вектор. При виконанні умови, що булеві функції нелінійні, не існує методу аналітичного отримання зворотного перетворення, тобто методу аналітичного розв'язання системи нелінійних булевих рівнянь. Основною перевагою використання булевих незворотних перетворень є саме те, що з їх використанням відкриваються можливості швидкої програмної та апаратної реалізації. Основним недоліком використання булевих незворотних перетворень є їх вузькі функціональні можливості, які не дозволяють створювати на їх основі складні криптографічні алгоритми.

Ще один клас математичних незворотних перетворень базується вже на використані теорії чисел. Класичний зразок незворотного перетворення представляє собою задача дискретного логарифмування. Доволі важливою перевагою цих перетворень є не тільки їх незворотність, але й неоднозначність – тобто наявність деякої множини зворотних перетворень, що ще на один рівень ускладнює злам таких алгоритмів та доступ до інформаційних повідомлень, які вони захищають. Саме ця властивість відкриває можливості побудови складних та ефективних криптографічно стійких конструкції на основі незворотних перетворень в теорії чисел: алгоритми несиметричного шифрування, криптографічні механізми

цифрового підпису та строгої ідентифікації на основі концепції “нульових знань”.

Виходячи з виду виконуваного криптографічного перетворення, усі криптографічні алгоритми діляться на дві наступні групи:

- зворотні: це ті, для яких алгоритмічно визначені процедури як прямого, так і зворотного перетворення, причому отримання зворотного перетворення є алгоритмічно нерозв'язним завданням (тобто завданням, яке в принципі може бути вирішне, проте тільки з використанням методів перебору);

- незворотні: ті, для яких алгоритмічно визначено тільки пряме перетворення, без зворотного, і результатом якого є цифрова сигнатура повідомлення.

Важливим класом зворотних алгоритмів є несиметричні криптографічні алгоритми, або як їх ще називають, алгоритми з відкритим ключем. Практично всі алгоритми що входять до цього класу, що використовуються на практиці, мають в своїй основі задачі з теорії чисел. Дуже важливою перевагою несиметричних алгоритмів у порівнянні з симетричними алгоритмами є те, що вони мають набагато більш широкі можливості створення на своїй основі ефективних протоколів для захисту інформації, які створюють передумову для дозволу відкритості одного з пари ключів (закриваючого або відкриваючого), але це можливо лише при умові, що другий ключ є завжди секретним(тобто його значення є лише у одного користувача). Виникнувши в кінці 70-х років криптографія з використанням відкритих ключів, породила за цей час велику кількість нових організаційних принципів захисту інформації, які багато дослідників [1] пов'язують з “новою епохою розвитку криптографії”.

1.2 Алгебра кінцевих полів Галуа і її використання в алгоритмах криптографічного захисту інформації

В сучасній криптографії, а також в теорії кодування, доволі широко та часто використовуються арифметичні операції, що відрізняються від звичайних тим, що при їх виконанні не виконуються дії міжрозрядних переносів.

Найбільш відомим класом таких операцій вважаються операції, що виконуються на полях Галуа [1]. Кінцеві поля Галуа $GF(2^n)$ – задаються множиною n-розрядних двійкових кодів, над якими задані операції додавання $\oplus$, множення $\otimes$ та відображення будь-якого числа довільної розрядності в поле.

Полем Галуа GF(2n) розмірності n називається обмежена множина дискретних елементів, на які задані операції додавання та множення, причому результати таких операцій, що виконуються над елементами полів Галуа, будуть завжди елементами полів Галуа, тобто визначені на полях Галуа елементи та операції ніколи не ніколи не вийде за межі поля. Зазвичай поле Галуа задається породжуючим поліномом виду [2]:

$$x^n = \sum_{j=0,\dots,n-1} a_j \cdot x^j \,. \qquad (1.1)$$

де $a_j \in \{0,1\}$, j=0,…,n-1. Для породження поля Галуа зазвичай використовують прості поліноми, які не можуть бути розкладені на множники. Тоді усі 2n елементи поля Галуа можуть бути представлені у вигляді наборів коефіцієнтів $a_j \in \{0,1\}$ при степенях x: $\{0, x^0, x^1, ..., x^{2^n-2}\}$ та навпаки: будь-якому n-розрядному двійковому числу завжди можливо поставити у відповідність поліном, який визначено на полі Галуа.

Так як в полі GF(2n) міститься скінчена кількість елементів, то в цьому полі найбільша система лінійно незалежних елементів $\beta_1,\beta_2,...,\beta_n$.

Відповідно, кожен елемент поля може бути виражено через вказані лінійно-незалежні елементи:

$$\forall \gamma \in GF(2^n): \gamma = c_1 \cdot \beta_1 \oplus c_2 \cdot \beta_2 \oplus ... \oplus c_n \cdot \beta_n,$$

де c_1, c_2,…,$c_n \in \{0,1\}$. Наприклад, якщо n=4 і вибрана ортогональна система елементів на відповідному полі: $\beta_1 = x+1$, $\beta_2 = x^3$, $\beta_3 = x^3+x^2$ та $\beta_4 = x^2+1$, то елемент цього поля $\gamma = x^3 + x^2 + x$, який співвідноситься з двійковим числом 1110 можна, згідно з (1.1) представити у вигляді лінійної комбінації від вибраної ортогональної системи: $\gamma = \beta_1 \oplus \beta_2 \oplus \beta_4$; відповідно: c_1=1, c_2=1, c_3=0, c_4=1. В цій ортогональній системі число γ=1110 трансформується до вигляду γ′=1011. Важливим є питання про те, скільки може бути утворено ортогональних систем елементів на кінцевому полі Галуа. Кількість варіантів вибору n-х елементів із 2^n-1 без повторень (нуль не входить до поля Галуа) становить $C^n_{2^n-1}$. Наприклад при n=4 існує $C_{15}^4 = 1365$ можливих систем із 4-х елементів поля, що не повторюються. Із них тільки 840 є ортогональними (близько 61%). При n=5 існує вже $C_{31}^5 = 169911$ систем з 5 –ти елементів, що не повторюються. З них ортогональних – 83328 (49%). Зі збільшенням n ймовірність того, що випадковим чином вибрана система виявиться ортогональною зменшується.

Базовим елементом поля Галуа є утворюючий поліном $P(x)$ степені n виду: $P(x) = x^n + b_{n-1}\cdot x^{n-1} + b_{n-2}\cdot x^{n-2}+\ldots+b_1\cdot x + b_0$, де коефіцієнти $b_0,b_1,\ldots,b_{n-1} \in \{0,1\}$. Утворюючий поліном поля $P(x)$ не може бути розкладений на множники, тобто він не може бути представлений у вигляді: $P(x) = (x^k \oplus z_k)\cdot Q(x)$ де $0<k<n-1$.

Кількість можливих утворюючих поліномів $N(n)$ залежить саме від степені n кінцевого поля Галуа і визначається за допомогою наступної формули [2]:

$$N(n) = \frac{\Phi(2^n - 1)}{n}\ , \qquad (1.2)$$

де $\Phi(2^n\text{-}1)$ –число Ейлера, тобто кількість чисел в інтервалі від 1 до 2^n-1, які не мають спільних подільників з числом 2^n-1. Наприклад, для $n = 4$ існує 8 чисел в інтервалі від 1 до 15: 1,2,4,7,8,11,13,14 які не мають спільних подільників з 2^4-1 = 15. Відповідно, для $n = 4$ $N(n) = 2$, тобто існує всього два простих поліноми степені 4: $P_1(x) = x^4 + x + 1$ та $P_2(x) = x^4+x^3+1$.

Поле Галуа GF(2^n) по суті являє собою мультиплікативну алгебраїчну групу, яка має порядок 2^n-1. Це означає, що для кожного елемента δ визначеного на полі, справедливо наступне: $\delta^{2^m-1} = 1$. З цього можливо зробити висновки, що мультиплікативна група елементів поля Галуа має властивість циклічності.

Поле Галуа GF(2^n) містить разом з кожним своїм елементом δ рівно один корінь 2-го степеня з δ. Важливою властивість поля Галуа є також і те, що добуток всіх його елементів дорівнює одиниці:

$$\prod_{j=1}^{2^n-1} j = 1$$

Операція додавання на полях Галуа співпадає з логічною побітовою операцією XOR, яка входить системи команд практично всіх процесорів та мікроконтролерів.

Операція множення на полях Галуа включає в собі дві складові: поліноміальне множення (множення без переносів) та редукцію отриманого результату, тобто віднаходження залишку від поліноміального ділення результату поліноміального множення на утворюючий поліном поля.

Добуток без переносів $D=\{d_{2\cdot q-1},\ldots,d_2,d_1\}$ двох q- чисел $W=\{w_q,\ldots,w_2,w_1\}$, та $V=\{v_n,\ldots,v_2,v_1\}$ обчислюється наступним чином:

$$D=V\cdot w_1 \oplus (V\cdot w_2)\cdot 2 \oplus (V\cdot w_3)\cdot 2^2 \oplus \ldots \oplus (V\cdot \omega_i)\cdot 2^{i-1} \oplus \ldots \oplus (V\cdot \omega_n)\cdot 2^{n-1}, \qquad (1.3)$$

де $\cdot$ – операція логічного множення.

Після операції поліноміального множення виконується операція редукції, тобто віднаходження залишку від ділення коду добутку на полі Галуа.

Традиційно ця операція виконується шляхом послідовного зсуву утворюючого поліному P(x) з додаванням його до $V(x)$. Кількість циклів таких операцій, в середньому, становить $n/2$. В кожному з циклів потрібно виконати n/r операцій зсуву (r-розрядність процесора), операцію тестування значення старшого розряду поліному, що редукується, а також n/r операцій додавання за модулем 2, якщо вказаний розряд дорівнює одиниці. В оціночному плані це потребує близько $1.5 \cdot n^2/r$ процесорних операцій.

1.3 Аналіз відомих способів експоненціювання на полях Галуа

Операція експоненціювання на кінцевих полях володіє двома важливими властивостями, які визначають можливість її ефективного використання в механізмах криптографічного захисту інформації:

1) В основі операції експоненціювання лежать дві обчислювальні процедури: множення без переносів (multiplication without carry), і редукція - тобто знаходження залишку від поліноміального розподілу результату множення на який утворює поліном поля P. Це множення виконується набагато простіше і швидше в порівнянні із звичайним арифметичним множенням. Аналогічно, операція поліноміального ділення виконується набагато простіше і швидше арифметичного ділення. Це дозволяє використовувати операцію множення на кінцевих полях в якості заходів щодо прискорення процедур криптографічного захисту даних в комп'ютерних системах та мережах.

2) В основі будь-якої криптографії лежить необоротне математичне перетворення, тобто таке перетворення, для якого принципово не може бути

аналітичним шляхом отримано зворотне. Класичними прикладами аналітично нерозв'язних завдань є дискретне логарифмування і рішення систем нелінійних булевих рівнянь. Цілком очевидно, що задача дискретного логарифмування нерозв'язна як в звичайній алгебрі, так і в алгебрі кінцевих полів. Практичним підтвердженням цього положення є використання подібних перетворень в стандарті блокового шифрування Rijndael . Очевидно, що єдиним способом вирішення задачі дискретного логарифмування як у звичайній алгебрі , так і в алгебрі кінцевих полів є перебір.

Щоб відрізняти операції на полях Галуа від аналогічних операцій, що виконуються у звичайній арифметиці, для позначення операції підсумовування нижче використовуються наступні позначення [3] . Операція підсумовування на полях Галуа, яка фактично відповідає логічному додаванню (XOR) нижче позначається символом ‘⊕’, тоді як символом '+' позначено звичайне додавання з переносами. Операція множення без міжрозрядних переносів нижче позначається знаком ‘⊗’ на відміну від звичайного множення, яке позначається символом '·'. Множення без міжрозрядних переносів $D=UV=\{d_{2\cdot n},\ldots,d_3,d_2,d_1\} = d_1+2d_2+4d_3+\ldots+2^{2r}d_{2r}$, $l\{1,..,2r\}$: $d_l\{0,1\}$ двох r-розрядних чисел $U=\{u_r,\ldots,u_2,u_1\}=u_1+2u_2+\ldots+2^r u_r$ та $V=\{v_r,\ldots,v_2,v_1\}= =v_1+2v_2+\ldots+2^r v_r$, $i\{1,..,r\}$: $v_i,u_i\{0,1\}$ обчислюється наступним чином: $D = Uv_1 \ (Uv_2)<<1 \ \ldots \ (Uv_r)<<(r\text{-}1)$, де $p << q$ - операція логічного зсуву числа p вліво на q розрядів. Операція піднесення числа A в ступінь E без переносів позначається як A^E. Операція обчислення залишку від ділення полінома, відповідного числу A на поліном, відповідний числу M позначена як A rem M. З урахуванням наведених позначень операція експоненціювання на полях Галуа позначена як A^E rem M, де M - число, відповідне утворюючому поліному поля.

Операція експоненціювання A^E на полі Галуа, що задається утворюючим його нерозкладним поліномом $Q(x)$ ступеня m припускає, що

всі його компоненти являють собою m-розрядні двійкові коди: $A=\{a_0, a_1,...,a_{m-1}\}$, $j\{0,...,m\text{-}1\}:a_j\{0,1\}$ та $E=\{e_0,e_1,...,e_{m-1}\}$, $e_j\{0,1\}$, яким відповідають поліноміальні подання $P(A)= a_0 + a_1x + a_2x^2 + a_3x^3 +...+a_{m-1}x^{(m-1)}$ та $P(E)= e_0 + e_1x + e_2x^2 + e_3x^3 +...+e_{m-1}x^{(m-1)}$.

До теперішнього часу запропоновано ряд способів виконання операції експоненціювання на полях Галуа [5-8]. Їх аналіз показує, що в якості основного резерву підвищення продуктивності їх автори розглядають можливість розпаралелювання виконання базової для експоненціювання операції множення на полях Галуа.

Сама процедура модулярного експоненціювання A^E rem M на полях Галуа, подібна до звичайного модулярного експоненціювання, зводиться до послідовного виконання m циклів, в кожному з яких здійснюється операція піднесення до квадрату отриманого на попередньому циклі результату і, додатково, в залежності від поточного біта експоненти E, e_h (нуль чи одиниця), виконується також операція множення. Виходячи з порядку, в якому аналізуються розряди експоненти E, існує два різновиди модулярного експоненціювання: справа – наліво і зліва – направо (на практиці частіше застосовується аналіз розрядів експоненти E починаючи зі старших розрядів, тобто зліва – направо) .

Формально цей спосіб може бути представлений у вигляді:

1. $R = 1$; $j = m\text{-}1$.

2. $R = (R \ R)$ rem M ;

3. if $e_j = 1$ then $R = (RA)$ rem M

4. $j = j\text{-}1$; if $j >= 0$ return to 2.

Результат $R= A^E$ rem M.

Оцінка обчислювальної складності наведеної процедури експоненціювання на полях Галуа може бути виконана через підрахунок необхідних операцій логічного підсумовування і зсувів. Процедура експоненціювання складається з m циклів, в кожному з яких виконується

операція зведення в квадрат на поле Галуа і, з імовірністю 0.5 - множення на постійне число A (рахуючи появу нулів і одиниць в коді експоненти рівноймовірно). Операція множення на полях Галуа складається з двох послідовно виконуваних фаз: поліноміального множення і редукції, яка зводиться до поліноміального поділу.

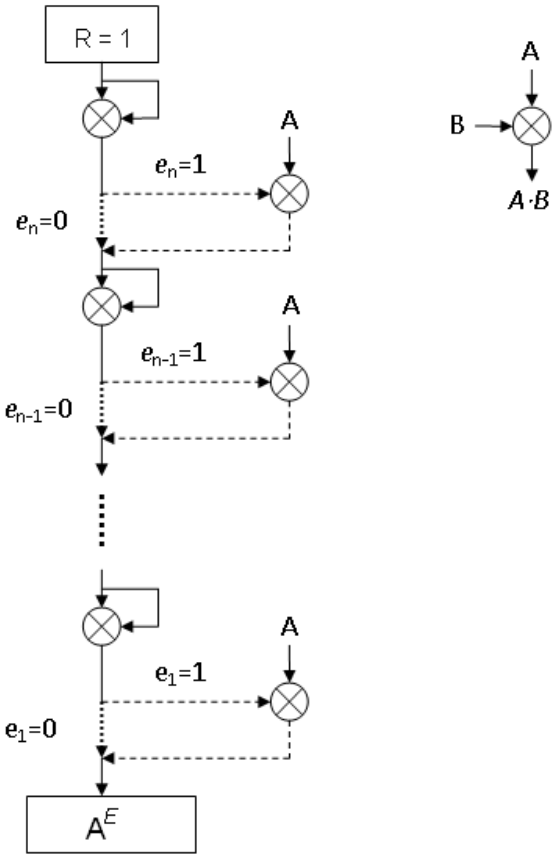


Рис. 1. 1 Структура операції експоненціювання

Поліноміальне множення, в свою чергу, також складається з m циклів, на кожному з яких з імовірністю 0.5 (залежно від значення поточного розряду множника) виконується логічне підсумовування множимого і проміжного результату, а також зсув коду множника (або проміжного результату). Таким чином, для виконання операції поліноміального множення, в середньому, потрібно $0.5 \cdot m$ операцій логічного підсумовування і m операцій зсувів, що в сумі складає $1.5 \cdot m$ елементарних операцій [6].

Аналогічно, операція поліноміального ділення $2 \cdot m$ розрядного коду на m- розрядний код утворюючого полінома вимагає m зрушень і, в середньому, $0.5 \cdot m$ операцій логічного підсумовування. Отже, операція множення на полях Галуа, вимагає, в середньому, $3 \cdot m$ елементарних операцій логічного складання і зсувів . Відповідно, операція експоненціювання на полях Галуа за описаним вище способом вимагає, в середньому, $Q = 4.5 \cdot m^2$ аналогічних операцій.

Найбільш відомим підходом до прискорення експоненціювання на полях Галуа є суміщення виконання операцій множення і ділення виду $u=ab/c$, де a, b, c - поліноми на полях Галуа. В основі підходу лежить запропонований Fitzpatrick P. [5] алгоритм поєднаного виконання операцій множення і ділення, який забезпечує знаходження шуканого результату $u=ab/c$ як мінімального елемента безлічі рішень тотожності. Практична реалізація цього підходу ефективна тільки в рамках апаратної реалізації експоненціювання .

Ефективність відомих методів прискорення експоненцірованія на полях Галуа значною мірою обмежується тим , що їх автори намагаються вирішити цю задачу в загальному вигляді, без урахування її специфіки для різних застосувань .

Висновки до розділу 1.

В результаті проведення досліджень, що складають перший розділ, і направлені на аналіз використання операції експоненціювання в сучасних криптографічних механізмах захисту інформації, огляду алгоритмів виконання цієї операції та аналізу існуючих методів її прискорення можна зробити наступні висновки:

1. Базовою обчислювальною операцією широкого кола алгоритмів криптографічного захисту інформації, зокрема криптографії на еліптичних кривих, є експонеціювання на кінцевих полях Галуа. Час виконання цієї операції вирішальним чином визначає швидкодію реалізації протоколів захисту інформації, в яких використовується алгебраїчний базис кінцевих полів Галуа.

2. Обидва класичних алгоритми експонеціювання на полях Галуа складається з рекурсивного виконання циклу, що складається з операції піднесення до квадрату на полях Галуа та множення на полях Галуа. Рекурсивний характер обох алгоритмів визначає строго послідовних характер їх обчислювальної реалізації. Показано, що алгоритм експоненціювання на кінцевих полях галуа з молодших розрядів коду експоненти дозволяє організувати два паралельних потоки обчислень.

3. Існуючі методи прискорення обчислення експоненти на кінцевих полях Галуа направлені, головним чином, на прискорення операцій піднесення до квадрату чи множення на полях Галуа. Ці операції можуть бути частково розпаралелені. Виходячи з того, що при використанні експоненціювання на полях Галуа в механізмах захисту інформації з відкритим ключем він є практично незмінним. Тому резервом прискорення експоненціювання на полях Галуа може бути використання результатів передобчислень, що залежать тільки від ключа.

4. Важливим резервом прискорення операції експоненціювання на кінцевих полях Галуа є зменшення часу виконання редукції проміжних результатів. В класичних алгоритмах операція редукції виконується після формування результату поліноміального множення. Для самої операції редукції проміжних результатів застосовується трудоємка операція поліноміального ділення. Існує можливість заміни операції поліноміального ділення більш ефективною, в обчислювальному плані, операцією. Крім того, для зменшення ресурсів процесору важливо виконувати редукцію після кожної операції додавання з тим, щоб обмежити зростання розрядності чисел, які потрібно оброблювати.

5. Вказані резерви прискорення операції експоненціювання на кінцевих полях Галуа можуть бути значною мірою реалізовані шляхом застосування технології Монтгомері. Така технологія успішно застосовується для прискорення редукції в традиційній алгебрі при виконанні модулярного експоненціювання. Відповідно задача дослідження полягає в адаптації технології Монтгомері для алгебри кінцевих полів Галуа та розробки методів редукції, множення та експоненціювання на кінцевих полях Галуа з використанням технології Монтгомері.

РОЗДІЛ 2

МЕТОД ПРИСКОРЕННЯ ЕКСПОНЕНЦІЮВАННЯ НА ПОЛЯХ ГАЛУА ЗА РАХУНОК ТЕХНОЛОГІЇ МОНТГОМЕРІ

Як було показано в попередньому розділі даної бакалаврської роботи, одним із перспективних та суттєво необхідних в дослідженнях напрямків прискорення виконання важливої для сучасних засобів захисту інформації операції експоненціювання на кінцевих полях Галуа є використання редукції Монтгомері. Відомо, що ця технологія широко використовується для прискорення виконання базової операції важливих криптографічних алгоритмів, таких як RSA, DSA, El-Gamal – модулярного експоненціювання. Задача досліджень, що проводяться в рамках поточного розділу бакалаврської роботи полягає в тому, щоб адаптувати відому технологію, розроблену Монтгомері, для алгебри кінцевих полів Галуа, розробити методику редукції з використанням зсуву замість більш складної і довшої в своїй апаратній реалізації операції поліноміального ділення, розробити метод множення на полях Галуа з використанням редукції на основі створеної технології Монтгомері.

На основі зробленого потрібно запропонувати нову розробку, метод експоненціювання на кінцевих полях Галуа з використанням редукції Монтгомері. Нарешті, потрібно виконати теоретичну та експериментальну оцінку ефективності експоненціювання на полях Галуа з використанням редукції Монтгомері в плані прискорення виконання цієї операції на різних обчислювальних платформах, при програмній та апаратній реалізації, аби довести можливість апаратного прискорення операції експоненціювання на полях Галуа, з використанням редукції Монтгомері, у порівнянні зі звичайним експоненціюванням на полях Галуа.

2.1. Аналіз обчислювальних процедур технології редукції Монтгоморі

В класичній алгебрі, для прискорення модулярного експоненціювання використовується технологія редукції Монтгомері. Відповідно до неї, усі обчислення виконуються не над самими числами, а лише над образами цих чисел, без редукції віднаходженням залишку, тому як операція знаходження залишку по модулю зводиться до виконання зсувів, це значно прискорює виконання усієї операції, адже складну та повільнішу у своєму апаратному виконанні операцію знаходження залишку по модулю замінюють операцію зсувів.

Отриманий результат модулярного експоненціювання трансформується з образного представлення у традиційне, за рахунок виконання певних перетворень над ним.

Основна ідея досліджень полягає в пристосуванні технології Монтгомері до експоненціювання на полях Галуа.

Операція редукції Монтгомері передбачає використання числа R, яке більше за модуль m і не має з ним спільних подільників: R > m gcd(R,m) = 1

Якщо є таке число x, яке менше за модуль m: $x<m$, то $x \cdot R^{-1} \bmod m$ – називається редукцією Монтгомері x по модулю m по відношенню до R.

R=32 m =19 R mod m = 32 mod 19 = 13 R-1 mod m = 3

При спеціальному виборі R, редукція Монтгомері зводиться до простої та швидкої у своїй реалізації операції зсуву.

Якщо попередньо обчислити x' = x·R mod m і y′ = y·R mod m, то при виконанні множення з рекурсією Монтгомері:

$$x' \cdot y' \cdot R^{-1} \bmod m = x \cdot R \cdot y \cdot R \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m = u'$$

Тобто фактично отриманий результат дорівнює:

$$u', u = x \cdot y \bmod m$$

Приклад:

$x=12 \quad x' = x \cdot R \bmod m = 12 \cdot 32 \bmod 19 = 4$

$y= 7 \quad y' = y \cdot R \bmod m = 7 \cdot 32 \bmod 19 = 15$

$x' \cdot y' \cdot R^{-1} \bmod m = x \cdot y \cdot R \bmod m$

$4 \cdot 15 \cdot 3 \bmod 19 = 9 = 12 \cdot 7 \cdot 32 \bmod 19$

Це означає, що при множенні трьох чисел можна перемножити образи Монтгомері двох перших і отриманий результат (в образі Монгомері) відразу помножити на образ третього співмножника

Приклад:

$x = 12 \quad x' = x \cdot R \bmod m = 12 \cdot 32 \bmod 19 = 4$

$y = 7 \quad y' = y \cdot R \bmod m = 7 \cdot 32 \bmod 19 = 15$

$z = 9 \quad z' = z \cdot R \bmod m = 9 \cdot 32 \bmod 19 = 3$

$x' \cdot y' \cdot R^{-1} \bmod m = 4 \cdot 15 \cdot 3 \bmod 19 = 9 = u'$

$u' \cdot z' \cdot R^{-1} \bmod m = 9 \cdot 3 \cdot 3 \bmod 19 = 5 = x \cdot y \cdot z \cdot R \bmod m = 12 \cdot 7 \cdot 9 \cdot 32 \bmod 19 = 5 = x' \cdot y' \cdot z' \cdot R^{-2} \bmod m = 4 \cdot 15 \cdot 3 \cdot 3 \cdot 3 \bmod 19 = 5$

На практиці, якщо модуль m складається з n двійкових розрядів, то число R вибирається як 2 в степені n, тобто: $R=2^n$. Тоді умови, за яких число R буде більше за модуль m, $R > m$ та $\gcd(R,m) = 1$ виконуються автоматично, оскільки, якщо модуль m – це просте число, або добуток двох простих чисел, то воно обов'язкового непарне.

Якщо $m' = R - m^{-1} \bmod R$ і x – довільне ціле число, для якого справедливо наступне: $0 \le x < m \cdot R$, то для числа U, обчисленого у вигляді $U = x \cdot m' \bmod R$ значення $x+U \cdot m$ націло ділиться на R, причому:

$(x+U \cdot m)/R = x \cdot R^{-1} \bmod m$.

Наприклад:

якщо m = 19, то $m^{-1} \bmod 32 = 27$ і, відповідно: $m' = 32 - 27 = 5$.

Для довільно обраного числа x, де x = 17, значення U: $U = x \cdot m' \bmod R = 17 \cdot 5 \bmod 32 = 21$. Тоді число $x + U \cdot m = 17 + 21 \cdot 19 = 416$. Воно націло ділиться на R=32: 416/32 = 13. Частка від ділення дорівнює $x \cdot R^{-1} \bmod m = 17 \cdot 3 \bmod 19 = 13$.

Таким чином, вузловим елементом ефективного застосування технології Монтгомері в традиційній модулярній арифметиці є вибір корегую чого числа, множення на яке дозволяє компенсувати некоректність результату через заміну операції ділення на зсув.

2.2 Аналіз спрощеного варіанту редукції Монтгомері

Для більш точного та глибокого розуміння операції редукції Монтгомері, яка покладена в основу розробленого методу прискорення експоненціювання на кінцевих полях, наведено ще один, спрощений варіант редукції Монтгомері.

За відповідних умов відомо, що існує *n*-бітовий модуль, такий, що $m=(m_{n-1},m_{n-2},\ldots,m_0)_2$, тоді значення $R=2^n$. Необхідно виконати редукцію такого числа $X=(x_{2n-1},x_{2n-2},\ldots,x_0)$, розрядність якого не більше ніж вдвоє перевищує розрядність модуля. На виході алгоритму формується значення $\eta = X \cdot R^{-1} \bmod m$.

1. for(j = 0, j < n; j++)
   1.1. if (x0 = 1) X = X + m
   1.2. X = X / 2

Приклад:

Нехай $m = 19_{10} = 10011_2$; n=5 і $R = 2^5 = 32$.

Потрібно виконати редукцію $X=59 = 111011_2$. На виході потрібно отримати значення $X \cdot R^{-1} \bmod m = 59 \cdot 3 \bmod 19 = 6$.

j=0 x0 = 1 X = X + m X=59 + 19 = 78 = 10011102

X = X/2 = 39 = 1001112

j=1 x0 = 1 X = X + m X= 39 +19 = 58 = 1110102

X = X/2 = 29 = 111012

j=2 x0 = 1 X = X + m X= 29 + 19 = 48 = 1100002

X = X/2 = 24 = 110002

j=3 x0 = 0 X = X/2 = 12 = 11002

j=4 x4 = 0 X = X/2 = 6 = 1102

2.3 Аналіз реалізації модулярного множення з застосуванням редукції Монтгомері

Вхідними даними є *n*-розрядні двійкові числа $A=(a_{n-1}\ a_{n-2}\ \dots\ a_1\ a_0\ )_2$, $B=(b_{n-1}\ b_{n-2}\ \dots\ b_1\ b_0\ )_2$ та модуль $M=(m_{n-1}\ m_{n-2}\ \dots\ m_1\ m_0)_2$. При виконанні операції множення використовується допоміжна величина $R=2^n$ та мультиплікативна інверсія R^{-1} цієї величини по модулю M, така, що $R\cdot R^{-1} \bmod M = 1$. Вихідною *n*-розрядною величиною $Y=(y_{n-1}\ y_{n-2}\ \dots.\ y_1\ y_0\ )$ є значення $A\cdot B\cdot R^{-1} \bmod M$.

Процедура множення Монтгомері складається за наступної послідовності дій:

1. Обнуляється поточний результат величини Y: Y = 0, лічильник j циклів (номер поточного двійкового розряду множника) встановлюється в нуль: j=0.

2. Обчислюється сума $Y = Y + b_j\cdot A$

3. Якщо значення поточного результату непарне, тобто $y_0 = 1$, то до поточного результату додається модуль Y = Y + M

4. Здійснюється зсув праворуч поточного результату: Y = Y / 2

5. Якщо j < n-1, то виконується інкремент *j*: *j*=*j*+1 і повернення на повторне виконання п. 2.

6. Якщо Y ≥ M, то Y = Y-M.

Функціонування описаної процедури може бути ілюстровано наступним прикладом.

Нехай дано модуль M = 10011_2 = 1910, відповідно, *n*=5; допоміжна величина R=2^5 = 32, а її мультиплікативна інверсія по модулю M дорівнює R^{-1} = 3. Виконується множення числа A=10001_2 = 17_{10} на число B= 01011_2 = 11_{10} .

Динаміка змін величин, що задіяні в процедурі виконання операції множення Монтгомері, в процесі покрокового виконання, проілюстровано нижче в таблиці 2.1

Таблиця 2.1

Динаміка покрокових змін величин, що задіяні в процедурі множення Монтгомері чисел A=17 і B=12

j	b_j	Поточний результат Y				
		На початку цикла	Після додавання множимого	y_0	Після додавання модуля	Після зсуву
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	1	0	0+17 = 17	1	17 + 19 = 36	36/2 = 18
3	1	18	18+17= 35	1	35 + 19 = 54	54/2 = 27
4	0	27	27	1	27 + 19 = 46	46/2 = 23
Та як Y > M (23 > 19), то згідно п.6: Y=Y-M = 23-19 = 4						

Отриманий результат Y=4 відповідає значенню A·B·R^{-1} mod M = 17·12·3 mod 19 = 4. Для отримання істинного значення A·B mod M результат Y можна помножити на R по модулю M: A·B mod M = Y· R mod M = 4·32 mod 19 = 14.

2.4 Модулярне експоненціювання з застосуванням редукції Монтгомері

Модулярне експоненціювання, доповнене з використанням редукції Монтгомері передбачає обчислення A^E mod M без операцій ділення для реалізації редукції. Число n дорівнює кількості двійкових розрядів модуля M. В процедурі модулярного експоненціювання використовується множення Монтгомері, яке позначається як MM(u,v) і яке формує модулярний добуток $u \cdot v \cdot R^{-1}$ mod M, де R^{-1} – мультиплікативна інверсія числа $R = 2^n$ по модулю M, тобто $R \cdot R^{-1}$ mod M=1. Код експоненти складається з h двійкових розрядів: $E=(e_h\ e_{h-1}\ \ldots.\ e_0\ )_2$, причому $e_h = 1$.

Перед виконанням самої процедури модулярного експоненціювання, здійснюються передобчислення, результати яких залежать тільки від незмінного впродовж усієї процедури модуля M: початкового значення поточного результату Q= R mod M, а також C= R^2 mod M.

Процедура модулярного експоненціювання з використанням рекурсії Монтгомері, тобто обчислення A^E mod M передбачає наступну послідовність дій:

1. Обчислюється значення B=MM(A,C).

2. Номер j поточного біту експоненти E встановлюється в h: j=h, так щоб він індексував старший одиничний двійковий розряд e_h коду експоненти

3. Здійснюється піднесення до квадрату Монтгомері поточного результату значення Q = MM(Q,Q).

4. Якщо поточний j-тий біт коду експоненти E дорівнює одиниці: e_j = 1, то виконується множення Монтгомері поточного результату на B: Q=MM(Q,B).

5. Якщо $j > 0$, то здійснюється інкремент j: $j=j$-1 і повернення на повторне виконання п.3.

6. Кінцевий результат отримується множенням Монтгомері отриманого значення Q на одиницю: Q = ММ(Q,1)

Описана процедура модулярного експоненціювання, що була модернізована та пришвидшена за рахунок використання редукції, розробленої Монтгомері, може бути ілюстрована наступним прикладом нижче.

Нехай модуль $M = 10011_2 = 19_{10}$, відповідно, n=5; допоміжна величина $R=2^5 = 32$, а її мультиплікативна інверсія по модулю M дорівнює $R^{-1} = 3$. Попередньо виконуються обчислення початкового значення поточного результату $Q = R \bmod M = 32 \bmod 19 = 13$, та значення C= $R^2 \bmod M = 32^2 \bmod 19 = 17$, які залежать тільки від незмінного модуля М.

При обчисленні $16^{11} \bmod 19 = 9$, значення А=16, код експоненти можна виразити як $E=11_{10} =1011_2 = (e_3\, e_2\, e_1\, e_0\,)_2$; тому можна отримати величину h: $h = 3$. У відповідності з п.1 послідовності дій при виконанні процедури модулярного експоненціювання з використанням рекурсії Монтгомері, обчислюється значення $R^2 \bmod M = 2^{64} \bmod 19 = 17$ та B = ММ(A, C) = ММ(16,17) = 18.

Динаміка зміни значень поточного результату по крокам виконання процедури модулярного експоненціювання Монтгомері проілюстрована в таблиці 2.2

Таблиця 2.2

Динаміка зміни поточного результату по крокам процедури модулярного експоненціювання Монтгомері при обчисленні $16^{11} \bmod 19$

j	e_j	Чисельне значення поточного результату Q		
		На початку циклу	Після піднесення до квадрату Монтгомері	Після множення Монтгомері на В
3	1	13	ММ(13,13) = 13	ММ(13,8) = 18

2	0	18	MM(18,18) = 3	3
1	1	3	MM(3,3) = 8	MM(8,18) = 14
0	1	14	MM(14,14) = 18	MM(18,18) = 3
Остаточний результат MM(3,1) = 9				

Процес модулярного експоненціювання за методом, запропонованим Монтгомері, можна теоретично пояснити на прикладі піднесення довільного числа A до степені 5 A^5 mod M.

Множення Монтгомері MM(u,v) формує модулярний добуток $u \cdot v \cdot R^{-1}$ mod M, де R^{-1} – мультиплікативна інверсія числа $R = 2^n$ по модулю M.

Початкове значення поточного результату $Q = R$ mod M. C= R^2 mod M. Обчислюється значення B яке дорівнює MM(A,C) = MM(A, R^2 mod M) = $A \cdot R^2 \cdot R^{-1}$ mod M = $A \cdot R$ mod M.

Оскільки код експоненти числа 5, E = 5_{10} = $(101)_2$,то відповідно h = 2.

На першому циклі при j=h=2 здійснюється множення Монтгомері MM(Q,Q), результат якого можна представити у вигляді $R \cdot R \cdot R^{-1}$ mod M = R mod M. Так як поточний розряд коду експоненти дорівнює одиниці: e_2=1, то на цьому циклі виконується множення Монтгомері MM(Q,B), результат якого можна представити у вигляді: (R mod M· A·R mod M · R^{-1}) mod M = A·R mod M.

На наступному циклі, при j=1 знов реалізується піднесення до квадрату Монтгомері MM(Q,Q) я результаті якого значення Q трансформується до вигляду: (A·R mod M· A·R mod M · R^{-1}) mod M = A^2·R mod M; оскільки e_1=0, то множення на B не виконується. На останньому циклі при j=0, здійснюється піднесення до квадрату Монтгомері: MM(Q,Q), результат якого можна представити у вигляді (A^2·R mod M· $A^2 \cdot R$ mod M · R^{-1}) mod M = A^4·R mod M. Так як e_0=1, то це значення множиться за Монтгомері на B:

MM(Q,B) з отриманням результату: ($A^4 \cdot R \bmod M \cdot A \cdot R \bmod M \cdot R^{-1}$) mod M = $A^5 \cdot R \bmod M$.

Після виконання трьох циклів реалізується множення Монтгомері отриманого значення Q на одиницю = MM(Q,1) = ($A^5 \cdot R \bmod M \cdot 1 \cdot R^{-1}$) mod $M = A^5 \bmod M$.

2.5 Розробка алгоритму редукції на полях Галуа на основі технології Монтгомері

На основі виконаний в попередніх підрозділах теоретичного аналізу використання технології Монтгомері для прискорення редукції в традиційній модулярній алгебри виявлено, що основним елементом, який забезпечує можливість застосування згаданої технології в іншій алгебрі, зокрема, в алгебрі кінцевих полів Галуа, є вибір корегуючого числа технології. Відповідно, задача полягає в теоретичному обґрунтування вибору аналогічного поліноміального представлення корегую чого числа для редукції на кінцевих полях Галуа.

Розібравшись с базовими принципами формування полів Галуа та їх кінцевих чисел, необхідно звернути увагу на наступну важливу дію, що виконується на цих полях.

В наш час існує багато різноманітних алгоритмів із різною сферою застосування. І одним із таких алгоритмів є A^E rem P(x) – експоненціювання на полях Галуа. Експоненціювання на полях Галуа – математичне перетворення, що є одним із основних механізмів захисту інформації. Його швидкість залежить від швидкості обчислення самої дії експоненціювання. А швидкість експоненціювання напряму залежить від розрядності чисел, які використовуються при обрахунках. Ще нещодавно це не було гострою проблемою, над якою треба було замислюватися. Але з появою хмарових технологій, доступ до суперкомп’ютерів став вільним. Звісно, технологія у руках звичайної людини не несе в собі ніякої загрози. Більшість людей

використовують їх здебільшого лише для зберігання інформації, навіть без отримання доступу до над комп’ютера чи можливих програмних ресурсів. Але у хмарових технологій є не лише ці можливості. Окрім доступу до пам’яті, вони надають доступ до ресурсів програмного забезпечення чи ресурсів обчислення самих суперкомп’ютерів. Маючи можливість використовувати ці ресурси, люди збільшують свій потенціал. Стає можливим вирішення низки важливих задач. Але разом з цим, хмарові технології мають і іншу, темну сторону. Саме ця темна сторона несе у собі загрозу для існуючих методів захисту інформації. В руках зловмисників технології можуть становити серйозну загрозу. Використовуючи суперкомп’ютер та його ресурси, будь-яка людина, звісно, маючи необхідні попередні навички, зможе отримати коди доступу, зламати банківські данні, або персональну інформацію інших людей, та ще багато іншого. Це призводить до того, що рівень захисту інформації падає, тому його силу необхідно збільшити. Але для цього необхідно збільшити й розрядність чисел, що використовуються при обчисленнях. І хоча це призведе до труднощів під час злому, навіть з використанням надкомп’ютерів. Раніше широко використовувалася числова розрядність 2048, проте, на сьогоднішній день, їй на зміну приходить інша, набагато більша. Збільшивши розрядність вдвічі, до 4096, збільшився і час обчислення. Причому не в два, а у вісім разів. Швидкість обчислення падає, та буде падати й надалі, зі збільшенням розрядності чисел. Саме тому, актуальною темою для криптографічних алгоритмів, направлених на захист інформації користувачів на сьогодення є прискорення швидкодії виконання операції експоненціювання на полях Галуа. Розробка цього методу забезпечить можливість підвищення розрядності без сильних витратах у часі, що безперечно зміцнить рівень криптографічної захищеності інформації та даних користувачів.

Таким чином, доведено, що в якості корегуючого поліному для застосування редукції на основі технології Монтгомері на кінцевих полях Галуа потрібно використовувати поліном виду $U(x)=x^n$, де n- ступінь утворюючого поліному поля Галуа. Іншими словами, корегуючий код для застосування технології Монтгомері на кінцевих полях Галуа залежить від утворюючого поліному поля тільки в рамках ступеню останнього. Тобто, для різних утворюючих поліномів поля Галуа, що мають однакову ступінь можна використовувати один і той же корегуючий поліном та його мультиплікативну інверсію.

Для досягнення поставленої мети – розробки швидкодіючого методу експоненціювання на полях Галуа, з використанням технології розробленої Монтгомері, необхідно попередньо визначити операцію редукції Монтгомері на полях Галуа. З застосуванням цієї редукції потрібно визначити операцію множення на полях Галуа.

Редукція Монтгомері на полях Галуа має замінити операцію більш складного та довшого в своїй апаратній реалізації операції поліноміального ділення. Для застосування редукції на полі Галуа, при заданому утворюючому поліномі $P(x)$ степені n потрібно здійснити редукцію поліному $V(x)$ степені не більшої $2\cdot n$: $V(x) = v_{2n-1}\cdot x^{2n-1} \oplus v_{2n-2}\cdot x^{2n-2} \oplus \ldots \oplus v_1\cdot x \oplus v_0$, $\forall l \in \{0,1,\ldots,2\cdot n\}: v_l \in \{0,1\}$. Відповідно, для цього обирається поліном $B(x)=x^n$. Для цього поліному визначається його мультиплікативна інверсія $B^{-1}(x)$ на полі Галуа, для якої виконується умова $B(x)\cdot B^{-1}(x) \text{ rem } P(x) = 1$. В результаті наведеної нижче процедури редукції утворюється поліном $V(x)\cdot B^{-1}(x) \text{ rem } P(x)$.

Процедура редукції передбачає наступну послідовність дій:

1. Лічильник j циклів встановлюється в нуль: $j=0$; створюється робоча копія $R(x)$ заданого поліному $V(x)$: $R(x) = V(x)$.

2. Якщо молодша компонента r_0 поліному $R(x)$ дорівнює одиниці: $r_0 = 1$, то до поліному $R(x)$ додається утворюючий поліном поля $P(x)$: $R(x) = R(x) \oplus P(x)$.

3. Здійснюється ділення поліному $R(x)$ на x: $R(x) = R(x)/x$

4. Якщо $j < n$, то виконується інкремент лічильника j: j=j+1 і повернення на повторне виконання п.2.

5. Кінець процедури рекурсії, результат якої знаходиться в R(x) : $R(x) = V(x) \cdot B^{-1}(x)$ rem P(x).

Приклад:

Процедура редукції Монтгомері на скінченному полі може бути ілюстрована наступним прикладом редукції поліному $V(x) = x^6 + x^5 + x^2 + x + 1$, який співвідноситься з числом $110111_2 = 55$ на полі Галуа, утвореним поліномом $P(x) = x^4 + x + 1$ (відповідне $P(x)$ число дорівнює $10011_2 = 19_{10}$). Відповідно, $n=4$. Для редукції вибирається допоміжний поліном $B(x) = x^4$, який співвідноситься з числом $10000_2 = 2^n = 16_{10}$, значення B^{-1} rem P = x^3+x^2+x, цьому поліному відповідає число$1110_2 = 14_{10}$.

Динаміка змін параметрів та змінних наведеної процедури відображена в таблиці 2.3.

Таблиця 2.3.

Динаміка змін поліномів та змінних при редукції поліному $V(x) = x^6 + x^5 + x^2 + x + 1$, який співвідностьс з числом $110111_2 = 55$

j	v_0	Числове значення поліному R(x)	
		При додаванні P(x)	Після ділення на x
0	1	$55 \oplus 19 = 110111 \oplus 10011 = 10010\mathbf{0}_2 = 36_{10}$	$36/2 = 18_{10} = 1001\mathbf{0}_2$
1	0	$1001\mathbf{0}_2$	$18/2 = 9_{10} = 100\mathbf{1}_2$
2	1	$9 \oplus 19 = 11010_2 = 26_{10}$	$26/2 = 13 = 110\mathbf{1}_2$
3	1	$13 \oplus 19 = 11110_2 = 30_{10}$	$30/2 = 15_{10} = 1111_2$

Таким чином, стає легко переконатися, що отримане значення 15_{10} відповідає значенню поліноміального добутку $V(x)\cdot B^{-1}(x)$ rem P(x) = 55·14 rem 19. Якщо це значення домножити на B(x)=16: R(x)·B(x) rem P(x) , то отриманий результат 15·16 rem 19 = 2 дорівнює значенню редукції поліному V(x) rem P(x) = 55 rem 19 =2.

Зазвичай, ця операція виконується шляхом послідовного зсуву утворюючого поліному P(x) з додаванням його до поліному $V(x)$. Кількість циклів таких операцій, в середньому, становить $n/2$, адже залежить від відповідності розряду нулю чи одиниці (на великих числах, з приведеною розрядністю в 2058 чи 4096 вірогідність нуля чи одиниці складає п'ятдесят відсотків). В кожному з циклів потрібно виконати n/r операцій зсуву (де r – розрядність процесора), операцію тестування значення старшого розряду поліному, що редукується, а також n/r операцій додавання за модулем 2, якщо вказаний розряд дорівнює одиниці. В оціночному плані це потребує близько $1.5\cdot n^2/r$ процесорних операцій.

При використанні редукції Монгомері виконується також n циклів, в кожному з яких здійснюється зсув, тобто складність навіть вища, тому, що потрібно зсувати $2\cdot n$- розрядний код V.

2.6 Розробка алгоритму множення на полях Галуа з застосуванням редукції Монтгомері

Вхідними даними для множення є два поліноми $C(x) = c_{m-1}\cdot x^{m-1} + c_{m-2}\cdot x^{m-2} +\ldots+ c_1\cdot x + c_0$ та $D(x) = d_{m-1}\cdot x^{m-1} + d_{m-2}\cdot x^{m-2} +\ldots+ d_1\cdot x + b_0$, $\forall i\in\{0,1,\ldots, m\text{-}1\}$: $c_j, d_j \in\{0,1\}$ ступеня m-1, з якими співвідносяться m-розрядні двійкові числа $C= c_{m-1}\cdot 2^{m-1} + c_{m-2}\cdot 2^{m-2} +\ldots+ c_1\cdot 2+ c_0$ та D = $d_{m-1}\cdot 2^{m-1} + d_{m-2}\cdot 2^{m-2} +\ldots+ d_1\cdot 2 + d_0$. Поле Галуа задається утворюючим поліномом Q(x) = $x^m + q_{m-1}\cdot x^{m-1} + q_{m-2}\cdot x^{n-2} +\ldots+ q_1\cdot x + p_0$ ступеню n, з утворюючим поліномом співвідноситься (n+1)-

розрядне двійкове число $P= 2^n + p_{n-1}\cdot 2^{n-1} + p_{n-2}\cdot 2^{n-2} +\ldots+ p_1\cdot 2 + p_0$.. При виконанні множення використовується допоміжний поліном $R(x) = x^n$, з яким співвідноситься число $R=2^{n-1}$. Для поліному може бути визначено поліном $R^{-1}(x)$ його мультиплікативної інверсії на полі Галуа, так, що $R(x)\cdot R^{-1}(x)$ rem $P(x) = 1$. Результатом множення Монтгомері на полі Галуа є поліном ступеня n-1 $Y(x) = y_{n-1}\cdot x^{n-1} + y_{n-2}\cdot x^{n-2}+ \ldots. + y_1\cdot x + y_0$), який співвідноситься з двійковим числом $Y= y_{n-1}\cdot 2^{n-1} + y_{n-2}\cdot 2^{n-2} +\ldots+ y_1\cdot 2 + y_0$ і співпадає з $A(x)\cdot B(x)\cdot R^{-1}(x)$ rem $P(x)$.

Процедура множення Монтгомері на полі Галуа складається за наступної послідовності дій:

1. Обнуляється поліном поточного результату $Y = 0$, лічильник j циклів встановлюється в нуль: $j=0$.

2. Обчислюється сума $Y(x) = Y(x) \oplus b_j\cdot A(x)$

3. Якщо $y_0 = 1$, то до поточного результату додається утворюючий поліном $Y(x) = Y(x) \oplus P(x)$.

4. Здійснюється операція зсуву праворуч поточного результату $Y(x)$: $Y(x) = Y(x)/x$

5. Якщо $j < n$-1, то виконується інкремент j: $j=j+1$ і повернення на повторне виконання п. 2.

Після зсуву, поточне значення Y завжди є поліномом ступеню n-1. Тому фінальну корекцію робити не потрібно.

Процедура може бути ілюстрована наступним прикладом множення на полі Галуа, утвореного поліномом $P(x) = x^4 + x+1$ (відповідне $P(x)$ число дорівнює $10011_2 = 19_{10}$). Для редукції вибирається допоміжний поліном $R(x) = x^4$, який співвідноситься з числом $10000_2 = 2^n = 16_{10}$, значення R^{-1} rem $P = x^3+x^2+x$, цьому поліному відповідає число $1110_2 =14_{10}$.

Виконується множення $A(x) =x^3 +x^2+x$ (відповідне число $A=1110_2 = 14_{10}$) на $B(x)= x^3 + 1$, який співвідноситься з числом $B= 1001_2 = 9_{10}$.

Динаміка змін величин, що задіяні в процедурі в процесі покрокового виконання відображена в таблиці 2.4.

Таблиця 2.4

Динаміка покрокових змін величин, що задіяні в процедурі множення Монтгомері на полі Галуа чисел A=14 і B=9

j	b_j	Поточний результат Y				
		На початку цикла	Після додавання множимого	y_0	Після додавання P	Після зсуву
0	1	0	$0 \oplus 14 = 14$	0	14	14 / 2 = 7
1	0	7	7	1	7⊕19 =20	20 / 2 = 10
2	0	10	10	0	10	10/2 = 5
3	1	5	5 ⊕14= 11	1	11⊕19 =24	24 / 2 = 12

Отриманий результат Y=10 відповідає значенню $A \cdot B \cdot R^{-1}$ rem M = 14·9·3 rem 19 = 12. Для отримання істинного значення A·B rem M = 14·9 rem 19 = 7 результат Y можна помножити на R по модулю M: A·B mod M = Y· R mod M = 12·16 rem 19 = 7.

Таким чином, доведено, що запропонований алгоритм множення на кінцевих полях Галуа дозволяє практично двоє прискорити реалізацію цієї базової операції за рахунок заміни операції поліноміального ділення на просту операцію зсуву, а також за рахунок зменшення розрядності чисел, що важливо для застосувань, в яких розрядність чисел значно перевищує розрядність процесору.

2.7 Розробка алгоритму прискореного експоненціювання на полях Галуа з застосуванням редукції Монтгомері

До цього в роботі було розглянуто вже існуюче експоненціювання на кінцевих полях Галуа. Для пришвидшення виконання цієї операції в

бакалаврській роботі пропонується використовувати операцію редукції Монтгомері, що зможе призвести до прискорення виконання дій. При можливості прискорення цієї важливої криптографічної операції для сучасних засобів захисту інформації, рівень захищеності інформації в сучасному світі зможе зрости, що гарантує і більший захист і користувачів, вододіючих цією інформацією.

Експоненціювання на полях Галуа з використанням редукції Монтгомері передбачає обчислення A^E rem P без операцій поліноміального ділення для реалізації редукції. Утворюючий поліном $P(x)$ степені n породжує поле n-розрядних чисел. В процедурі експоненціювання на полях Галуа використовується поліноміальне множення на полі Галуа з застосуванням редукції Монтгомері, яке позначається як MG(u,v) і формує добуток $u \cdot v \cdot R^{-1}$ rem P, де R^{-1} – мультиплікативна інверсія $R(x)=x^n$ на полі Галуа, утвореного поліномом P(x). Код експоненти складається з h двійкових розрядів: $E=(e_h\ e_{h-1}\ \ldots.\ e_0\ )_2$, причому, перший розряд коду експоненти завжди має дорівнювати одиниці, (тобто $e_h = 1$), адже нулем можливо було б знехтувати, що приведе до зміни поточної розрядності з h до $h-1$.

Попередньо, здійснюється обчислення початкового значення поточного результату $O = R$ rem P(x) = x^n rem $P(x) = P(x) \oplus x^n$, та C = $R \cdot R$ rem $P = x^{2 \cdot n}$ rem $P(x) = x^{2 \cdot n}$ rem $P(x)$, значення яких залежать тільки від утворюючого поліному $P(x)$ поля Галуа, а отже і не будуть змінюватися впродовж виконання операції, тому ці дії необхідно виконати лише один раз, перед початком.

Процедура експоненціювання на полях Галуа з використанням рекурсії Монтгомері, тобто обчислення A^E rem P передбачає наступну послідовність дій:

1. Обчислюється значення B = MG(A,C).

2. Номер j поточного біту експоненти E встановлюється в h: $j=h$, так щоб він індексував старший одиничний двійковий розряд e_h коду експоненти.

3. Здійснюється операція піднесення до квадрату на кінцевому полі Галуа з використанням редукції Монтгомері поточного результату*:* O = MG(O,O)

4. Якщо поточний j-тий біт коду експоненти Е дорівнює одиниці: e_j = 1, поточний результат множиться на В на полі Галуа з редукцією Монтгомері : OP= MG(O,B).

5. Якщо j >0, то здійснюється інкремент j: $j=j$-1 і повернення на повторне виконання п.3.

6. Кінцевий результат отримується множенням отриманого значення O поточного результату на полі Галуа з використанням редукції Монтгомері на одиницю: $O = MG$ (O,1).

Описана процедура експоненціювання на полях Галуа з використанням редукції Монтгомері може бути проілюстрована з використанням наступного прикладу.

Якщо утворюючий поліном поля Галуа має вигляд $P(x) = x^4 + x+1$, тобто співвідноситься з числом P=19, то n=4 і, відповідно, $R(x) = x^4$. Початкове значення результату – O, що залежить лише від утворюючого поліному поля Галуа, попередньо обчислюється у вигляді: $O(x) = R(x)$ rem $P(x) = x+1=3$. Також обчислюється значення $C(x) = R^2(x)$ rem $P(x) = x^{2\cdot n}$ rem P = x^2+1 або в числовій формі С = 5.

При обчисленні $(x^3 +1)^5$ rem $P(x)$, що в числовій формі відповідають значенням 9^5 rem 19= 7, значення А=9 і код експоненти дорівнює Е = 101_2 = 5_{10} тобто h=2.

Згідно п.1 обчислюється $B(x)$= MG(A,C) = MG(9,5) = 8.

Динаміка зміни значень поточного результату по крокам виконання процедури експоненціювання показана в таблиці 2.5.

Таблиця 2.5.
Динаміка зміни поточного результату по крокам процедури експоненціювання на полях Галуа

j	e_j	Чисельне значення поточного результату		
		На початку циклу	Після піднесення до квадрату з рекурсією Монтгомері	Після множення з рекурсією Монтгомері на B
2	1	3	MG(3,3) = 3	MG(3,8) = 8
1	0	8	MG(8,8) = 4	4
0	1	4	MG(4,4) = 1	MG(1,8) = 9
Остаточний результат MG(9,1) = 7				

9^5 rem 19 = 7

Відповідно до цього, можна побачити, що результатом роботи описаної вище процедури є дійсно формування в O значення A^E rem P =
$= A^{e_h \cdot 2^h + e_{h-1} \cdot 2^{h-1} + \ldots + e_1 \cdot 2 + e_0}$ rem P.

В основі процедури лежить операція поліномільного множення Монтгомері на полі Галуа – MG(u,v), яка формує добуток $u \cdot v \cdot R^{-1}$ rem P, де R^{-1} – мультиплікативна інверсія $R(x)=x^n$ на полі Галуа, утвореного поліномом P(x), тобто:

$$\mathrm{MG}(u,v) = R(x) \cdot R^{-1}(x) \text{ rem } P(x) = 1.$$

Попередньо, здійснюється обчислення залежних лише від утворюючого поліному $P(x)$ поля Галуа значень: початкового значення поточного результату $O = R$ rem P та параметра C = R^2 rem P.

При обчисленні A^E rem P = $A^{e_h \cdot 2^h + e_{h-1} \cdot 2^{h-1} + \ldots + e_1 \cdot 2 + e_0}$ rem P згідно з наведеною вище процедурою, виконується h циклів, в яких послідовно, починаючі зі старших, тестуються бітові значення коду експоненти. В кожному циклі поточний результат O підноситься до квадрату з редукцією Монтгомері і, при одиничному значенні поточного біту коду експоненти помножається з редукцією Монтгомері на число B. Якщо позначити через i порядковий

номер циклу виконання процедури, то i змінюється від нуля до h, а прийняте в описі процедури значення індексу j пов'язано з i як $j=h-i$.

На нульовому циклі ($i=0$, $j=h$ і $e_j = 1$) початкове значення числа O дорівнює $O = R \text{ rem } P$, поліноміальне піднесення його до квадрату з використанням редукції Монтгомері не змінює значення O: $O = \text{MG}(O \cdot O) = O \cdot O \cdot R^{-1} \text{ rem } P = R \cdot R \cdot R^{-1} \text{ rem } P = R \text{ rem } P$. Оскільки $e_j = 1$, здійснюється множення на полі Галуа з редукцією Монтгомері O на $B = A \cdot R \text{ rem } P$: $O=\text{MG}(O \cdot B) = O \cdot B \cdot R^{-1} \text{ rem } P = (R \text{ rem } P \cdot (A \cdot R) \text{ rem } P \cdot R^{-1}) \text{ rem } P = R \cdot A \cdot R \cdot R^{-1} \text{ rem } P = A \cdot R \text{ rem } P$.

На першому циклі ($i=1$, $j=h-1$) значення $O = A \cdot R \text{ rem } P$ підноситься до квадрату: $O = \text{MG}(O \cdot O) = O \cdot O \cdot R^{-1} \text{ rem } P = (A \cdot R) \cdot (A \cdot R) \cdot R^{-1} \text{ rem } P = \text{A}^2 \cdot R \text{ rem } P$. Якщо $e_{h-1}=1$, то значення O множиться з редукцією Монтгомері на $B = A \cdot R \text{ rem } P$ з отриманням результату: $O=\text{MG}(O \cdot B) = O \cdot B \cdot R^{-1} \text{ rem } P = (\text{A}^2 \cdot R \text{ rem } P \cdot (A \cdot R) \text{ rem } P \cdot R^{-1}) \text{ rem } P = A^2 \cdot R \cdot A \cdot R \cdot R^{-1} \text{ rem } P = A^{2+1} \cdot R \text{ rem } P$. Якщо розряд коду експоненти дорівнює нулю, тобто $e_{h-1}=0$, то значення O не змінюється і дорівнює $\text{A}^2 \cdot R \text{ rem } P$. Таким чином, після виконання першого циклу, значення проміжного результату O має дорівнювати наступному значенню: $A^{2+e_{h-1}} \cdot R \text{ rem P}$.

Аналогічним чином, можна дійти до висновку також і про те, що після і-го циклу значення проміжного результату O визначається наступним виразом:

$$O_i = A^{2^i + e_{h-1} \cdot 2^{i-1} + \ldots + e_{h-i+1} \cdot 2 + e_{h-i}} \cdot R \quad rem\, P .$$

Після виконання всіх $h+1$ циклів при $i=h$ значення Q визначається такою формулою, при $i=h$:

$$O_h = A^{2^h + e_{h-1} \cdot 2^{h-1} + \ldots + e_1 \cdot 2 + e_0} \cdot R \quad rem\, P$$

Після виконання всіх циклів отримане значення O_h множиться на одиницю на полях Галуа з рекурсією Монтгомері:

$$O = MG(O_h, 1) = (A^{2^h + e_{h-1} \cdot 2^{h-1} + \ldots + e_1 \cdot 2 + e_0}) \cdot R \cdot 1 \cdot R^{-1} \; rem P = A^E \; rem P$$

2.8 Оцінка ефективності

При виконанні класичного алгоритму експоненціювання на полях Галуа виконується $1.5 \cdot m$ операцій поліноміального множення. В запропонованому методі також використовується $1.5 \cdot m+1$ операцій поліноміального множення з редукцією Монтгомері. Враховуючи, що на практиці значення m становить 2048 або 4096, можна вважати, що класична та запропонована процедури експоненціювання на полях Галуа використовують приблизно однакову кількість операцій поліноміального множення - $1.5 \cdot m$.

Операція полільного множення m-розрядних чисел потребує для обчислення добутку 0.5·m операція логічного додавання і m операцій зсуву. Така ж кількість операцій потрібна для виконання поліноміального ділення, через яку реалізується обчислення редукції добутку.

Беручи до уваги, що час виконання команди логічного додання приблизно однаковий з часом виконання команди зсуву, можна вважати, що реалізація поліноміального множення визначається часом виконання $3 \cdot m$ логічних операцій.

Запропонована організація виконання поліноміального множення з редукцією Монтгомері складається з m циклів, на кожному з яких, в середньому, одна операція логічного додавання (XOR) і одна операція зсуву. Відповідно, реалізація операції поліноміального множення з редукцією Монтгомері потребує $2 \cdot m$ логічних операцій.

Це означає, що запропонована організація експоненціювання на полях Галуа потребує $3 \cdot m^2$ логічних операцій, в той час, як класична – $4.5 \cdot m^2$.

Таким чином, організація виконання поліноміального множення з редукцією Монтгомері виконується в 1.5 раз швидше в порівнянні з традиційною операцією множення на полях Галуа. Проведені

експериментальні дослідження, в цілому, підтвердили наведені теоретичні оцінки прискорення обчислення експоненти на полях Галуа.

Запропонована організація прискореного експоненціювання більш ефективна в порівнянні з відомими методами, зокрема, основаними на перед обчисленнях, які забезпечують прискорення всього лише в 1.2 рази.

Таким чином, в результаті проведених теоретичних досліджень в бакалаврському дипломі запропоновано метод прискорення експоненціювання на кінцевих полях Галуа, який відрізняється тим, що для поліноміальної редукції проміжних результатів множення на піднесення до квадрату використовується технологія Монтгомері, застосування якої відоме в традиційній модулярній арифметиці, і яка дозволяє замінити ресурсоємку операцію поліноміального ділення операцію зсуву, що забезпечує значне зниження обчислювальної складності редукції проміжних результатів обчислень і, відповідно, прискорення експоненціювання на кінцевих полях Галуа. В рамках створення методу прискореного експоненціювання на кінцевих полях Галуа здійснене теоретичне обґрунтування і запропоновано процедури редукції на кінцевих полях з використанням технології Монтгомері, процедури множення на кінцевих полях з використанням редукції Монтгомері і власне сама процедура експоненціювання на полях Галуа зі спрощеною редукцією.

Основним фактором досягнутого ефекту – прискорення обчислювальної реалізації операції експоненціювання на полях Галуа з використанням процедури Монтгомері є можливість роботи з операндами вдвічі меншої розрядності, що за реальної ситуації коли довжина операндів значно переважає розрядність процесора забезпечує прискорення обчислень практично вдвоє. Найбільший ефект в плані прискорення обчислення експоненти на кінцевих полях Галуа запропонований метод експоненціювання на полях Галуа з використанням процедури Монтгомері

досягається при апаратній реалізації цієї операції в складі криптопроцесорів та крипто прискорювачів.

Висновки до розділу 2

В результаті здійснення досліджень, які складають поточний другий розділ дипломного проекту і націлені на розробку методу редукції на полях Галуа з застосуванням технології Монтгомері і на цій основі алгоритмів множення та експоненціювання на полях Галуа, можна зробити такі висновки:

1. На основі теоретичного аналізу використання технології Монтгомері для прискорення редукції в традиційній модулярній алгебри виявлено, що основним елементом, який забезпечує можливість застосування згаданої технології в іншій алгебрі, зокрема, в алгебрі кінцевих полів Галуа, є вибір корегуючого числа технології. Теоретично обґрунтовано вибір поліноміаль¬ного представлення відповідного числа для редукції на кінцевих полях Галуа.

2. На основі теоретичних досліджень запропоновано алгоритм редукції на кінцевих полях Галуа з використанням технології Монтгомері, який дозволяє замінити ресурсоємку операцію поліноміального ділення на більш просту в реалізації операцію арифметичного зсуву праворуч. Це дозволило прискорити виконання операції редукції на полях Галуа.

3. На основі розробленого алгоритму на полях Галуа запропоновано алгоритм множення на кінцевих полях Галуа з застосуванням редукції на основі технології Монтгомері. Доведено, що запропонований алгоритм множення на кінцевих полях Галуа дозволяє практично двоє прискорити реалізацію цієї базової операції за рахунок заміни операції поліноміального ділення на просту операцію зсуву, а також за рахунок зменшення розрядності чисел, що важливо для застосувань, в яких розрядність чисел значно перевищує розрядність процесору.

4. Теоретично обґрунтовано, розроблено та досліджено алгоритм експоненціювання на кінцевих полях Галуа з застосуванням редукції на основі технології Монтгомері. Теоретично доведено, що запропонований

алгоритм забезпечує прискорення виконання цієї важливої для сучасних протоколів криптографічного захисту інформації обчислювальної операції. Прискорення залежить від розрядності чисел: чим більше розрядність, тим більший виграш у швидкодії.

5. В порівнянні з іншими відомими метолами прискорення обчислювальної реалізації експоненціювання на кінцевих полях Галуа, запропонований підхід, оснований на адаптації до особливостей алгебри кінцевих полів відомої технології Монтгомері, забезпечує більший виграш у швидкодії та менший об’єм потрібних для цього ресурсів, зокрема на відміну від методу передобчислень, запропонований не потребує витрат пам'яті для зберігання таблиць передобчислень.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ МЕТОДУ ОБЧИСЛЕННЯ ЕКСПОНЕНТИ НА ПОЛЯХ ГАЛУА З ВИКОРИСТАННЯМ РЕДУКЦІЇ МОНТГОМЕРІ

Сучасні технології програмування зумовили наявність в більшості мов програмування апаратної чи запрограмованої на низькому рівні більшості операції з вбудованими типами даних. Зокрема, цілочисельний тип Integer має розрядність 32 біта і дорівнює розрядності більшості процесорів. Це дозволяє здійснювати всі арифметичні та логічні операції безпосередньо на процесорному, тобто апаратному рівні.

В криптографічних алгоритмах захисту інформації застосовуються числа з розрядністю, значно більшою за розрядність процесора. Так, наприклад, для стандартів цифрового підпису на сьогодні використовуються цілі числа розрядністю 2048. Це зумовлено тим, що заданий рівень захищеності, який вимірюється об’ємом ресурсів, потрібних для зламу захисту, досягається лише при великій розрядності чисел, що утруднює порушення захисту, яке, в більшості випадків зводиться до факторізації числа, тобто розкладання його на множники. Робота з типами цілих чисел, довжина яких значно перевищує розрядність процесора, базується на ідеї представлення цих чисел у вигляді бітового рядка або масиву чисел, кожне з яких відповідає розрядності процесора і містить в собі частину бітів (32 або 64) повного числа. Найчастіше використовується для представлення таких довгих чисел одновимірний масив з двох елементів типу Integer. Арифметичні операції над числами цього типу виконуються над кожним фрагментом окремо з використанням спеціальних процедур об'єднання фрагментарних результатів.

Такий підхід роботи з чисельними типами, довжина котрих перевищує

розрядність процесору, ускладнює виконання операцій над числами великої розрядності, що використовуються в реальних протоколах захисту інформації. З іншого боку при використанні такого підходу можна підвищити швидкодію виконання арифметичних операцій, якщо вони на апаратному рівні виконуються процесором (наприклад множення). Ускладнює ситуацію те, що потрібно реалізувати формування та поширення переносу з молодших розрядів у старші, а це має наслідком уповільнення виконання операцій, ускладнює або навіть унеможливлює розпаралелювання виконання операції задачі на багатоядерних процесорах.

Програмну реалізацію запропонованого методу прискорення операції експоненціювання на кінцевих полях Галуа за рахунок застосування для редукції проміжних результатів технології Монтгомері здійснено на мові програмування Пітон. Вибір мови програмування зумовлений тим, що це забезпечує найменший час створення програми, легкість її відлагоджування, простоту внесення змін. Для розробки, основною ціллю якої є експериментальне дослідження нового методу з можливістю змін структури перетворень таким вибір вбачається оптимальним. Мова програмування Пітон достатньо нова в порівнянні з іншими мовами програмування типу C++ чи Jawa вона має вбудовані засоби роботи з багато розрядними числами, в тому числі і тими, розрядність яких значно перевищує розрядність процесора. Це суттєвим чином спрощує програмування. Більше того, засоби мови Пітон дозволяють гнучко міняти розрядність чисел, що дуже зручно для дослідження залежності параметрів швидкодії від довжини чисел. Таким чином, використання мови програмування Пітон дозволяє швидко и просто написати програму для експериментального дослідження розробленого методу прискорення операцій експоненціювання на кінцевих полях Галуа. Ця програма добре підходить для отримання порівняльних оцінок швидкодії різних структурних рішень. З використанням цієї програми можна доволі просто експериментально порівняти ефективність

запропонованого методу в порівнянні з іншими існуючими підходами до прискорення обчислення експоненти на полях Галуа. Для коректного порівняння і аналізу результатів швидкодії необхідно використати однакові принципи представлення чисел і роботи з ними, та не використовувати реалізовані в мові типи, для цього необхідно реалізувати ще й операції в класичній арифметиці.

Разом з тим, використання мови програмування Пітон не дозоляє отримати оптимальний в плані досягнення максимальної швидкодії програмний код експоненціювання на кінцевих полях Галуа. Наявність вбудованих засобі не дозволяє використати специфічні умови задач, пов’язаних з захистом інформації. наприклад, експериментально доведено, що час виконання множення двох чисел та піднесення до квадрату двох чисел в Пітоні виконується за практично однаковий час. Разом з тим, добре відомо, що при оптимізації виконання піднесення до квадрату, ця операція виконується для багато розрядних чисел практично вдвічі швидше. При розробці мови програмування Пітон піднесення до квадрату вважалося рідкою операцією, для реалізації якої використано вбудовану операцію множення багато розрядних чисел. Тому в мові Пітон час піднесення до квадрату та множення практично не відрізняється. Разом з тим, в алгоритмах криптографічного захисту інформації операція піднесення до квадрату є домінуючою: її питома вага для більшості криптографічних алгоритмів з відкритим ключем складає близько 72%. Тому при реалізації таких алгоритмів реально потрібно спеціально писати процедуру піднесення до квадрату.

Як і іншу мови програмування Пітон не пристосований для виконання операцій в кінцевих полях Галуа: він не має вбудованих операцій поліноміального множення та поліноміального ділення. Тому ці операції оформлені у вигляді спеціально розроблених програмних модулів.

Для представлення чисел візьмемо за основу двійкову систему, числа

будуть представлені у вигляді одновимірного масиву булевих змінних, де true – це одиниця, false – нуль. Таке представлення цілих беззнакових чисел реалізовано за допомогою розробленого типу Num. Над змінними цього типу можна виконувати арифметичні операції в двох алгебрах.

Першою з цих двох алгебр є класична арифметика в якій враховується перенос з молодших розрядів числа в старші при виконнні операції додавання. Операції над нею реалізовані над об'єктами типу Num. Другою алгеброю є модифікована арифметика полів Галуа, що використовується в запропонованому методі прискореного експоненціювання на кінцевих полях. При розрахунку чергового розряду, в операції множення, за основу взято операцію виключного АБО (далі просто XOR) замість класичного додавання. В цьому випадку перенос з молодших розрядів не враховується і, відповідно, не поширюється на старші. Такі операції також реалізовано над змінними типу Num. Метою програми, що розроблюється рамках поточного розділу бакалаврської роботи є порівняльний аналіз часу виконання експоненціювання на полях Галуа та запропонованого описаного в попередньому розділі роботи алгоритму прискорення експоненціювання на полях Галуа з використанням редукції Монтгомері.

3.1 Організація представлення даних

Для представлення даних в розробленій програмі використовується вбудований тип *poly*(*n*), де n – кількість членів поліноміального представлення двійкового число, яке практично збігається з розрядністю цього двійкового числа. В програмі використовуються поліноми, які мають дві довжини – n та $2 \cdot n$. Перші поліноми використовуються для представлення чисел нормальної розрядності, на яку розрахований алгоритм захисту інформації з відкритим ключем, я якому використовується операція експоненціювання на кінцевих полях Галуа. Поліноми другого типу

використовуються для роботи з проміжними результатами мультиплікативних операцій на полях Галуа, які мають збільшену удвічі розрядність. Реально тип, що використовуються в розроблені програмі співпадає з бітовим рядком.

В програмі, яка реалізує безпосереднє експоненціювання на полях Галуа з застосуванням редукції Монтгомері не використовується складних структур даних. Проте складні структури даних використано для порівняльного аналізу запропонованого методу і способів прискорення експоненціювання на полях Галуа що мають за основу використання результатів передобчислень, що залежать лише від утворюючого поля Галуа, яке являється частиною відкритого ключа криптографічних алгоритмів і, відповідно, практично не змінюється протягом тривалого часу. Для зберігання результатів передобчислень використано спеціальні таблиці поліномів, які являють собою результат ділення поліномів виду x^{n+1}, x^{n+2}, x^{n+3},…,$x^{2\cdot n}$ на утворюючий поліном кінцевого поля Галуа $P(x) = x^n + a_{n-1}\cdot x^{n-1} + a_{n-2}\cdot x^{n-2}+\ldots+a_1\cdot x + 1$. Очевидно, що в таблиці T зберігається n об’єктів типу $poly(n)$.

Для прямого і зворотного перетворення двійкових чисел типу Num в поліноміальне представлення типу $poly(n)$ використовуються вбудовані функції перетворення типів. При виводі робочих змінних програми виводяться лише коефіцієнти поліноміального представлення чисел.

Доцільність поліноміального представлення чисел диктується, насамперед тим, що розроблена програм оперує саме з поліномами різного ступенями. Крім того, вбудовані операції над поліноміальним типом дозволяє напряму звертатися до окремих бітів рядка поліноміальних коефіцієнтів.

3.2. Розробка програмних модулів

Для досягнення високої гнучкості при проведенні експериментальних досліджень запропонованого методу прискореного обчислення експоненти на полях Галуа в програмі використовуються чоти базових методи.

Команда множення на кінцевих полях Галуа з використанням редукції Монтгомері montgomery_mul_desc(*na*, *nb*, *np*). Ця команда здійснює поліноміальне множення *na* на *nb* без формування проміжного результату перед його редукцією. Особливістю виконання операціє є те, що результат поліноміального множення не формується в повному обсязі, тобто у вигляді $2 \cdot n$ – розрядного коду. В програмі часткова редукція здійснюється відразу після обробки одного розряду множника, тому відсутнє накопичення проміжного результату. Фактично, розрядність проміжного результату в застосованій схемі множення не перевищує $n+1$.

Таке технологічне рішення дозволяє зменшити кількість бітових операцій більш ніж в 5 раз. Це досягається за рахунок того, що при множенні на полях Галуа за розробленим алгоритмом не накопичується всі $2 \cdot n$ розрядів результату. Відповідно, обробка проміжного результату здійснюється не в форматі $2 \cdot n$, а у двічі меншому, тобто в n-розрядному форматі поліному.

Якщо переданий в якості аргументу функції поліном *na* множника не містить елементів, то функція повертає зміну result, яка являє собою результуючий поліном операції множення. Якщо ж поліном множника не дорівнює Nil, тоді відбувається перевірка розрядів полінома, починаючи з молодших, якщо поточне значення бітового рядка множника *nb* дорівнює true (одиниця), то здійснюється операція XOR між поліномом множеного і результатом попередньої ітерації, інакше кажуче, елемент рівний false, результат лишається не змінним. Прохід по всім розрядам полінома множника здійснюється через рекурсивний виклик функції з новими

значенням проміжних змінних, отриманими на поточному кроці ітерації. В якості першого значення проміжного результату передається поточний поліном множеного зсунутий вліво на один розряд. Другим параметром функції виступає поліном множника без обробленого на поточному кроці розряду, третім параметром є бітовий рядок утворюючого поліному поля.

В якості множника R в програмі використовується значення $2|^{n+1}$ rem $P(x)$. В результаті множення утворюється бітовий рядок, який фактично не є коректним результатом множення на полів Галуа: для отримання правильного результату потрібно домножити отриманий в результаті множення бітовий рядок на корегуючий поліном.

Крім того, в процедурі множення не використовується ресурсоємкий пошук старшої значущої одиниці проміжного результату та порівняння його з зсунутим кодом утворюючого поліному коду Галуа – $P(x)$.

Другою розробленою базовою командою є процедура піднесення до квадрату на кінцевих полях Галуа def montgomery_sqr_desc(*na, np*). Ця команда має лише два параметри: перший параметр – *na* – бітовий рядок, який співвідносится з числом, яке підноситься до квадрату. Другий параметр – це бітовий рядок *np*, якій співвідноситься з утворюючим поліномом поля Галуа. Як зазначалося вище, операція піднесення до квадрату може виконуватися на полях Галуа за спрощеною схемою.

Для ефективної реалізації операції експоненціювання на полях Галуа ключову роль відіграє специфічна властивість поліноміального квадрату, а саме: двійкові розряди поліноміального квадрату числа C, що знаходяться на парних позиціях дорівнюють нулю, а біти, які локалізуються на непарних розрядах квадрату співпадають з бітами коефіцієнтів поліноміального представлення числа C, тобто, якщо $C=c_0+c_1\cdot x+c_2\cdot x^2+...+c_{n-1}\cdot x^{n-1}$, то $C\otimes C=C|^2 = c_0+c_1\cdot x^2+c_2\cdot x^4+...+c_{n-1}\cdot x^{2\cdot n-2}$. Наприклад, якщо $n=6$ і $C=111011_2$, то $C\otimes C = 1010100010_2$. Найважливіше, з точки зору організації обчислень, полягає в тому, що обчислення поліноміального квадрату (без редукції) не потребує

для своєї реалізації обчислюваних операцій і зводиться лише до вставки нульових бітів в рядок представлення поліному проміжного результату.

Обчислення поліноміального квадрату $C/^2$ не потребує спеціальних є обчислювальних ресурсів для своєї реалізації і в практичному плані зводиться до вставки нулів між розрядами числа С. Це відкриває можливості для об'єднання операції поліноміального піднесення в квадрату і поліноміальне множення на постійне число, з яких складається операція експоненціювання на кінцевих полях Галуа. Сама операція піднесення до квадрату на полях Галуа включає себе дві складові: саму операцію поліноміального піднесення до квадрату, яка зводиться до вставки нулів між розрядами проміжного результату і редукцію отриманого квадрату, тобто віднаходження залишку від поліноміального ділення квадрату на утворюючий поліном кінцевого поля Галуа. Результат піднесення до квадрату записується у бітовий набір result всі розряди якого перед початком циклу множення встановлюються в нуль. Формування квадрату поліному виконується шляхом організації циклу від 0 до значення ступеню утворюючого поліному поля Галуа. Підчас кожної ітерації циклу формування квадрату на полі Галуа після поточного біту числа додається нульовий біт.

Технологічно вставка нулів в код проміжного результату С може виконуватися двома способами. Перший спосіб передбачає послідовну процедуру сканування бітового рядка С і вставки в рядок результату нуля (на парних номерах циклу) або поточного біту рядка С зі зсувом останнього на непарних номерах циклів. Зрозуміло, що операція в цьому варіанті займає $2 \cdot n$ циклів. Очевидний недолік цього варіанту реалізації поліноміального піднесення до квадрату полягає в низькій швидкодії, яка зумовлена необхідністю виконувати $3 \cdot n$ операцій зсуву коду.

Другий варіант полягає в тому, що зсуву на різну кількість позицій використовуються таблиці передобчислень. Якщо одночасно

оброблюються k розрядів проміжного результату, то таблиця передобчислень має містити 2^k рядків, в кожному з яких зберігаються бітові вагові коефіцієнти часткового квадрату. Зрозуміло, що число таких вагових коефіцієнтів дорівнює $2 \cdot k$. Тоді для обчислення з використанням раніше сформованих таблиць потрібно n/k операцій звернення до таблиці та зсуву.

Третьою базовою командою розробленої програми є власне процедура експоненціювання на кінцевих полях Галуа з використанням редукції Монтгомері *montgomery*_exp(*a*, *ne*, *p*). В якості параметрів цієї процедури виступають: число a, над яким здійснюється операція експоненціювання, число *ne* – ступінь в яку підноситься число *a*, або просто експонента. Третім параметром процедури є *p* –утворюючий поліном кінцевого поля Галуа, на якому здійснюється операція експоненціювання.

Сама операція експоненціювання, яка реалізується в розробленій програмі передбачає попереднє обчислення допоміжних змінних, які використовуються під час редукції Монтгомері. Допоміжна *n*-бітова допоміжна змінна *u_m* обчислюється як функція від заданого параметру *p* – утворюючого поліному кінцевого поля Галуа, на якому здійснюється операція експоненціювання. Для того, щоб, згідно технології Монтгомері, можна було звести ресурсоємку операцію поліноміального ділення накопиченого $2 \cdot n$-бітового проміжного результату простого зсуву праворуч, передбаченого процедурою редукції Монтгомері потрібно попередньо, тобто до основного циклу експоненціювання, сформувати змінну *u_m*, яка співвідноситься з поліномом ступеню *n* $W(x) = x^n$. Фактично число *u_m* складається з однієї одиниці і *n*-1 нулів. Після визначення змінної *u_m* формується змінна *m_u* яка є мультиплікативною інверсією *u_m*.

Для віднаходження мультиплікативної інверсії на заданому утворюючим поліномом полі Галуа в розробленій програмі використано алгоритм Евкліда. Суть цього алгоритму зводиться до виконання послідовних кроків знаходження найбільшого спільного подільника цілих

чисел великої розрядності. Для знаходження мультиплікативної інверсії в розробленій програмі шляхом послідовного віднаходження залишків використовується $\log_2 n$ циклів.

В розробленій програмі використовується різновид алгоритму експоненціювання на полях Галуа зі старших розрядів коду експоненти. Це дозволяє спростити програму і залучити менші ресурси пам'яті, оскільки множення на полях Галуа здійснюється на один і той же множник. В виконаній розробці постійний елемент множення відіграє роль множимого, що дозволяє скоротити час виконання програми за рахунок попереднього об'єднання двох операцій в одну: в пам'яті зберігається множиме і мультиплікативна інверсія множимого. На кожному циклі алгоритму виконується здійснюється логічне додавання до проміжного результату або множимого, або його мультиплікативної інверсії.

В розробленій програмі організовано виконання всіх n циклів експонеціювання на полях Галуа. В принципі, розглядалася сожливість переривання операції, якщо розряди експоненти, що ще не обраблені дорівнюють нулю. Це дозволить, в середньому, скоротити 1.86 циклів (тобто 2.45 операцій множення на полях Галуа), проте буде мати наслідком помітне ускладнення програми. Зрозуміло, при при виконанні більше тисячі циклів таке скорочення (на 0.2%) не дає помітного виграшу в швидкодії експоненціювання на полях Галуа.

Для виконання піднесення до квадрату на полі Галуа використовується описана вище процедура montgomery_sqr_desc(*na*, *np*), а для множення на постійне число - процедура множення на полі Галуа з застосуванням редукції Монтгомері.

Процедури exp_desc та mul_desc організують можливість виконання алгоритмів експоненціювання на множення на полях Галуа відповідно в покроковому режимі.

3.3 Експериментальні дослідження

Розроблена програма може функціонувати в двох режимах: дослідницькому та моделюючому.

В дослідницькому режимі користувач має змогу від слідкувати логіку експоненціювання на полях Галуа з використанням редукції Монтгомері, ввести різни числа для того, щоб побачити роботу алгоритму і програми в критичних режимах. В цьому режимі можна змінювати параметри операції експоненціювання на полях Галуа, вводити різні вхідні дані.

Наявність цього режиму дозволяє виявити різні конфліктні ситуації, наприклад побачити, як буде працювати алгоритм у випадку вводу не коректних даних, зокрема в ситуаціях коди утворюючий поліном не є простим, тобто він може бути представлений у вигляді поліноміального добутку двох інших поліномів. Очевидно, що в цій ситуації поле Галуа утворює не одне, а декілька кілець. Це може мати наслідком неможливість віднаходження мультиплікативної інверсії. Розроблена програма в цій ситуації має вбудовані засоби контролю циклів виконання алгоритму Евкліда: якщо алгоритм не сходиться за $\log_2 n$ циклів, то здійснюється вихід з циклу в режимі переривання з виводом повідомлення. Аналогічні ситуації можуть виникати коли задаче число або код експоненти мають ступінь більшу, ніж утворюючий поліном поля Галуа.

На рис.3.1 показано, в якості прикладу, вікно інтерфейсу розробленої програми в дослідницькому режимі. Також на рис.3.1 показана можливість роботи програми в покроковому режимі з виводом на екран значень всіх проміжних змінних та поточних результатів.

В режимі моделювання здійснюється статистичне моделювання розробленої процедури експоненціювання на полях Галуа з застосуванням редукції Монтгомері. При цьому утворюючий поліном вводиться в інтерактивно, а інші параметри генеруються випадковим чином, але так, щоб задовольняти умові бути меншим за утворюючий поліном поля.

```
cmd:>exp_desc
a:9
e:5
p:19
initial values:
  q = 3
  c = 5
step: 2
  after squaring:
    y = 3
  e[j] == 1, y = 3 * 8
    y = 8
step: 1
  after squaring:
    y = 4
step: 0
  after squaring:
    y = 1
  e[j] == 1, y = 1 * 8
    y = 9
last step:
  y = y*1
    y = 7
cmd:>mul_desc
a:14
b:9
p:19
starting calculation...
step 0:  b[0] == 1, adding a
    y = [0, 1, 1, 1, 0] or 14
  dividing by 2
    y = [0, 0, 1, 1, 1] or 7
step 1:  y[0] == 1, adding p
    y = [1, 0, 1, 0, 0] or 20
  dividing by 2
    y = [0, 1, 0, 1, 0] or 10
step 2:  dividing by 2
    y = [0, 0, 1, 0, 1] or 5
step 3:  b[3] == 1, adding a
    y = [0, 1, 0, 1, 1] or 11
  y[0] == 1, adding p
    y = [1, 1, 0, 0, 0] or 24
  dividing by 2
    y = [0, 1, 1, 0, 0] or 12
A*B*R^(-1) rem M = 12 (4 binary digits)

cmd:>
```

Рис.3.1 Приклад роботи програми в дослідницькому режимі

Робота в режимі моделювання потрібна для експериментального дослідження часових характеристик роботи алгоритму експоненціювання на полях Галуа з редукцією Монтгомері, проведення порівняльного аналізу з іншими методами прискорення цієї важливої для практики криптографічного захисту інформації операції, встановлення залежності показників швидкодії від ступеню утворюючого поліному.

Зокрема, з використанням розробленої програми експериментально показано, що ефект прискорення виконання експоненціювання на полях Галуа за рахунок використання редукції Монтгомері зростає зі збільшенням

розрядності чисел. При розрядностях близьких до 2048, що є зараз стандартом для більшості протоколів криптографічного захисту, застосування редукції Монтгомері дозволяє більш ніж удвічі прискорити експоненціювання на полях Галуа.

Висновки до розділу 3

В результаті здійснення досліджень та розробок, які складають поточний розділ дипломного проекту і націлені на створення програмних засобів реалізації прискореного обчислення експоненти на полях Галуа з застосуванням редукції Монтгомері та експериментальну оцінку ефективності використання технологій Монтгомері в криптографічних алгоритмах на основ кінцевих полів Галуа отримані наступні результати:

1. Розроблені та протестовані програмні засоби експоненціювання на полях Галуа з застосуванням технології Монтгомері швидкої редукції проміжних обчислень, які практично реалізують запропоновані в другому розділі дипломного проекту алгоритми редукції на полі Галуа з використанням технології Монтгомері, алгоритму множення на полях Монтгомері з редукцією за методом Монтгомері та алгоритму експоненціювання на полях кінцевих Галуа з редукцією Монтгомері. Розробка програмних засобів здійснена на мові програмування Пітон. Розроблені програми орієнтовані для роботи в двох режимах – дослідницькому та статистичного моделювання.

2. За допомогою розроблених програмних засобів експериментально доведено правильність та коректність проведених теоретичних досліджень, показана на практиці коректність роботи запропонованих алгоритмів на полях Галуа, які використовують редукцію Монтгомері. Експериментально досліджено функціонування розроблених алгоритмів в критичних режимах.

3. Проведені з використанням розроблених програм експериментальні дослідження ефективності запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції довели на практиці, що ці алгоритми забезпечують збільшення швидкодії в 1.5-2.5 рази в залежності від розрядності в порівнянні з класичним алгоритмом експоненціювання на полях Галуа.

4. Аналіз одержаних експериментальним шляхом результатів показав, що ефективність запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції зростає при збільшенні розрядності чисел.

5. Проведені з використанням розроблених програмних засобів експериментальні дослідження запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції підтверджують, в цілому, отримані в другому розділі теоретичні оцінки.

ВИСНОВКИ

В результаті проведених в рамках дипломного проекту досліджень, направлених на вирішення актуальної для сьогодення наукової задачі прискорення операції експоненціювання на полях Галуа, яка лежить в основі широкого кола алгоритмів криптографічного захисту інформації, за рахунок застосування технології Монтгомері швидкої редукції проміжних обчислень отримані наступні результати:

1. Базовою обчислювальною операцією широкого кола алгоритмів криптографічного захисту інформації, зокрема криптографії на еліптичних кривих, є експонеціювання на кінцевих полях Галуа. Час виконання цієї операції вирішальним чином визначає швидкодію реалізації протоколів захисту інформації, в яких використовується алгебраїчний базис кінцевих полів Галуа.

2. Обидва класичних алгоритми експонеціювання на полях Галуа складається з рекурсивного виконання циклу, що складається з операції піднесення до квадрату на полях Галуа та множення на полях Галуа. Рекурсивний характер обох алгоритмів визначає строго послідовних характер їх обчислювальної реалізації. Показано, що алгоритм експоненціювання на кінцевих полях галуа з молодших розрядів коду експоненти дозволяє організувати два паралельних потоки обчислень.

3. Існуючі методи прискорення обчислення експоненти на кінцевих полях Галуа направлені, головним чином, на прискорення операцій піднесення до квадрату чи множення на полях Галуа. Ці операції можуть бути частково розпаралелені. Виходячи з того, що при використанні експоненціювання на полях Галуа в механізмах захисту інформації з відкритим ключем він є практично незмінним. Тому резервом прискорення експоненціювання на полях Галуа може бути використання результатів передобчислень, що залежать тільки від ключа.

4. Важливим резервом прискорення операції експоненціювання на кінцевих полях Галуа є зменшення часу виконання редукції проміжних результатів. В класичних алгоритмах операція редукції виконується після формування результату поліноміального множення. Для самої операції редукції проміжних результатів застосовується трудоємка операція поліноміального ділення. Існує можливість заміни операції поліноміального ділення більш ефективною, в обчислювальному плані, операцією. Крім того, для зменшення ресурсів процесору важливо виконувати редукцію після кожної операції додавання з тим, щоб обмежити зростання розрядності чисел, які потрібно оброблювати.

5. Вказані резерви прискорення операції експоненціювання на кінцевих полях Галуа можуть бути значною мірою реалізовані шляхом застосування технології Монтгомері. Така технологія успішно застосовується для прискорення редукції в традиційній алгебрі при виконанні модулярного експоненціювання. Відповідно задача дослідження полягає в адаптації технології Монтгомері для алгебри кінцевих полів Галуа та розробки методів редукції, множення та експоненціювання на кінцевих полях Галуа з використанням технології Монтгомері.

6. На основі теоретичного аналізу використання технології Монтгомері для прискорення редукції в традиційній модулярній алгебри виявлено, що основним елементом, який забезпечує можливість застосування згаданої технології в іншій алгебрі, зокрема, в алгебрі кінцевих полів Галуа, є вибір корегуючого числа технології. Теоретично обґрунтовано вибір поліноміального представлення відповідного числа для редукції на кінцевих полях Галуа.

7. На основі теоретичних досліджень запропоновано алгоритм редукції на кінцевих полях Галуа з використанням технології Монтгомері, який дозволяє замінити ресурсоємку операцію поліноміального ділення на більш

просту в реалізації операцію арифметичного зсуву праворуч. Це дозволило прискорити виконання операції редукції на полях Галуа.

8. На основі розробленого алгоритму на полях Галуа запропоновано алгоритм множення на кінцевих полях Галуа з застосуванням редукції на основі технології Монтгомері. Доведено, що запропонований алгоритм множення на кінцевих полях Галуа дозволяє практично двоє прискорити реалізацію цієї базової операції за рахунок заміни операції поліноміального ділення на просту операцію зсуву, а також за рахунок зменшення розрядності чисел, що важливо для застосувань, в яких розрядність чисел значно перевищує розрядність процесору.

9. Теоретично обґрунтовано, розроблено та досліджено алгоритм експоненціювання на кінцевих полях Галуа з застосуванням редукції на основі технології Монтгомері. Теоретично доведено, що запропонований алгоритм забезпечує прискорення виконання цієї важливої для сучасних протоколів криптографічного захисту інформації обчислювальної операції. Прискорення залежить від розрядності чисел: чим більше розрядність, тим більший виграш у швидкодії.

10. В порівнянні з іншими відомими метолами прискорення обчислювальної реалізації експоненціювання на кінцевих полях Галуа, запропонований підхід, оснований на адаптації до особливостей алгебри кінцевих полів відомої технології Монтгомері, забезпечує більший виграш у швидкодії та менший об’єм потрібних для цього ресурсів, зокрема на відміну від методу передобчислень, запропонований не потребує витрат пам'яті для зберігання таблиць передобчислень.

11. Розроблені та протестовані програмні засоби експоненціювання на полях Галуа з застосуванням технології Монтгомері швидкої редукції проміжних обчис¬лень, які практично реалізують запропоновані в другому розділі дипломного проекту алгоритми редукції на полі Галуа з використанням технології Монтгомері, алгоритму множення на полях

Монтгомері з редукцією за методом Монтгомері та алгоритму експоненціювання на полях кінцевих Галуа з редукцією Монтгомері. Розробка програмних засобів здійснена на мові програмування Пітон. Розроблені програми орієнтовані для роботи в двох режимах – дослідницькому та статистичного моделювання.

12. За допомогою розроблених програмних засобів експериментально доведено правильність та коректність проведених теоретичних досліджень, показана на практиці коректність роботи запропонованих алгоритмів на полях Галуа, які використовують редукцію Монтгомері. Експериментально досліджено функціонування розроблених алгоритмів в критичних режимах.

13. Проведені з використанням розроблених програм експериментальні дослідження ефективності запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції довели на практиці, що ці алгоритми забезпечують збільшення швидкодії в 1.5-2.5 рази в залежності від розрядності в порівнянні з класичним алгоритмом експоненціювання на полях Галуа.

14. Аналіз одержаних експериментальним шляхом результатів показав, що ефективність запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції зростає при збільшенні розрядності чисел.

15. Проведені з використанням розроблених програмних засобів експериментальні дослідження запропонованих алгоритмів експоненціювання на кінцевих полях Галуа з застосуванням технології Монтгомері для прискореної редукції підтверджують, в цілому, отримані в другому розділі теоретичні оцінки.

СПИСОК ЛІТЕРАТУРИ

1. Зензин О. С. Стандарт криптографической защиты XXI века – AES. Конечные поля./ О.С. Зензин, М.А. Иванов //М.: Кудиц-Образ. — 2002 — 174 с.
2. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер //М.:”Триумф”. — 2002. – 816 с.
3. Николайчук Я.М. Коди полів Галуа: теорія і застосування / Я.М. Николйчук //Тернопіль.-Вид-во ТНУ. —2012. - 576 с.
4. Марковський О.П. Використання алгебри полів Галуа для реалізації концепції «нульових знань» при ідентифікації та автентифікації віддалених / О.П.Марковський, Захаріудакіс Ліфтеріс, Максимук В.Р. // Електронне моделювання. Збірник наукових праць. - 2017.- Т.6, №39. - С.33-45.
5. Francis N. Closed formulas for exponential sums of symmetric polynomials over Galois fields / N.Francis, A.Luis, M. Castro // Journal of Algebraic Combinatorics, — 2019. — V. 50. - P. 73-98.
6. Рахма, М. Структурна складність помножувачів елементів полів Галуа у нормальному та поліноміальному базисах / М. Рахма, Р. Еліас, В. Глухов // Електротехнічні та комп’ютерні системи. – Одеса: – 2017. Вид-во Наука і техніка. - № 25.- С. 332-340.
7. Марковський О.П. Технологія цифрового підпису DSA на основі арифметики полів Галуа / О.П. Марковський, Саїдреза Махмали, Ісаченко Г.В. // Вісник націо¬нального технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. — 2012. - № 55. - С. 34 — 41.
8. Саидреза Махмали. Реализация схемы идентификации FFSIS на основе умножения без переносов/ Саидреза Махмали // Вісник націо-нального

технічного універ¬ситету України ”КПІ”. Інформатика, управління та обчислювальна техніка. — 2011. - № 54.- С. 62-70.

9. Калмиков І.А. Розробка методу нелінійного шифрування інформації з використанням операції піднесення до степеня для кінцевого поля Галуа / І.А. Калмиков, Е.С. Степанова, К.Т. Тинчеров// Сучасні наукомісткі технології. — 2019.- № 9.- 2019. - С.84—89.
10. Анісімов А.В. Алгоритмічна теорія великих чисел / А.В. Анісімов // К.: Академперіодика. —2001. — С.153.
11. Fitzpatrick P. Algorithm and Architecture for a Galois Fiels multiplicative Arithmetic Processor./ P. Fitzpatrick, Popovici E. M. // IEEE Trans. on Information Theory. — 2003— V.49, - № 12, — P. 3303— 3307.
12. Wu H. Finite field multiplier using redundant representation./ H. Wu, M.A. Hasan, I.F. Blake, S.Gao // IEEE Trans. Computers. —2002 — V.51,- № 51. — P. 1306 — 1316.
13. Постников М.М. Теория Галуа / М.М.Постников // СПб.: Изд-во “БХВ-Петрербург”.— 2011. — С. 411.
14. Самофалов К.Г. Эффективная реализация мультипликативных операций модулярной арифметики в системах защиты информации/ К.Г. Самофалов, Г.М.Луцкий, А.П. Марковский // Proceeding of International scientific conference UNITECH-09. Gabrovo 20-21 November 2009.- Technical University of Gabrovo. - 2009.— V.1.— P.435-437.
15. Стефанская В.А. К проблеме повышения эффективности аппаратной реализации мультипли¬кативных операций на полях Галуа. / В.А. Стефанская, Мухаммад Мефлех Алиса Абабне, Д.Ю. Левчун // Вісник Національ¬ного техніч¬ного університету України ”КПІ”. Інформатика, управління та обчислюваль¬на техніка. К.,"ВЕК++", - 2005. - № 43.- С.104-112.

16. Самофалов К.Г. Способ ускоренной реализации экспоненцирования на полях Галуа в системах защиты информации / К.Г. Самофалов, А.П.Марковський, А.С. Шаршаков // Проблеми інформа¬тизації та управління. Збірник наукових праць: К.,НАУ.- 2011. - Т.2, № 33 - С.143-151.

17. Харин Ю.С. Математические основы криптологии./ Ю.С.Харин, В.И. Берник, Г.В. Матвеев//Минск.: Изд-во БГУ —1999. — 319 с.

18. Montgomery P.L. Modular multiplication without trial division / P. L. Montgomery // Mathematics of Computation, - 1985.- Vol.44. - № 4.- P.519-521.

19. Osadchyy V. The Order of Edwards and Montgomery Curves «, / V.Osadchyy // WSEAS Transactions on Mathematics, - 2020.- Vol. 19.- № 25, - P. 253-264.

20. Elfard S. Justification of Montgomery Modular Reductions / S. Elfard //Advanced Computing.- 2012. - № 11. – P.41-45.

21. Haches G. Montgomery multiplication with no final subtraction./ G. Haches, J.J. Quisquater // Cryptographic Hardware and Embedded System-CHES’2000. LNCS-1965, Springer-Verlag. — 2000.- P. 293-301

22. Марковский О.П. Метод строгої автентифікації віддалених абонентів на основі алгебри полів Галуа / Максимук В.Р. //Міжнародна науково-практична конференція ICSFTI2018. Київ 10-12 Травня 2018.- КПІ ім. Ігоря Сікорського. — 2018. — Р.355— 365.

23. Кот О.С. Організація прискореного експоненціювання на полях Галуа з використанням редукції Монтгомері / О.С.Кот, О.П. Марковський // Альманах науки.- 2020.- № 3 (36).- С.34-37.

24. Markovskyi Oleksandr The Employment of Montgomery reduction for acceleration of exponent on Galoise fields calculation / O. Markovskyi, V.

Masimyk, O.Kot // Proceeding of International Conference on Security, Fault Tolerance, Intelligence” (ICSFTI2020), 13-14 травня 2020, м. Київ.- P.44-49.

Додатки

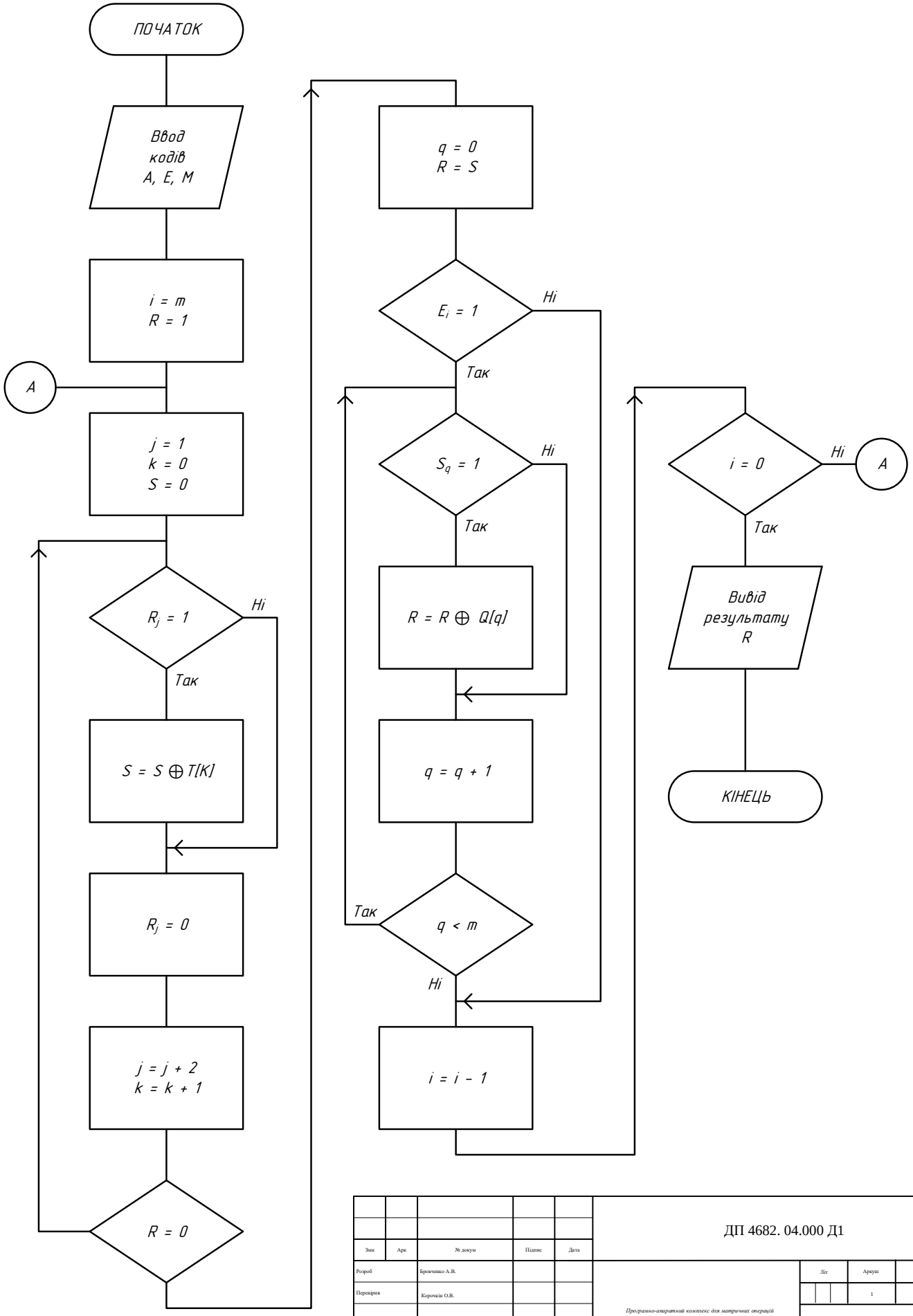
ДОДАТОК А

Метод та програмні засоби прискореного експоненціювання на полях Галуа

Схема алгоритму класу

Аркушів 1

Київ – 2020 року



Зми	Арк	№ докум	Підпис	Дата
Розроб		Брюховецко А.В.		
Перевірив		Корочкін О.В.		
Н. Контр		Симоненко В.П.		
Затверд.				

ДП 4682. 04.000 Д1

Програмно-апаратний комплекс для матричних операцій
Схема принципова топології ПАК.

Літ	Аркуш	Аркушів
	1	1

ІТУУ «КПІ» ФІОТ
ГРУПА ІО-64

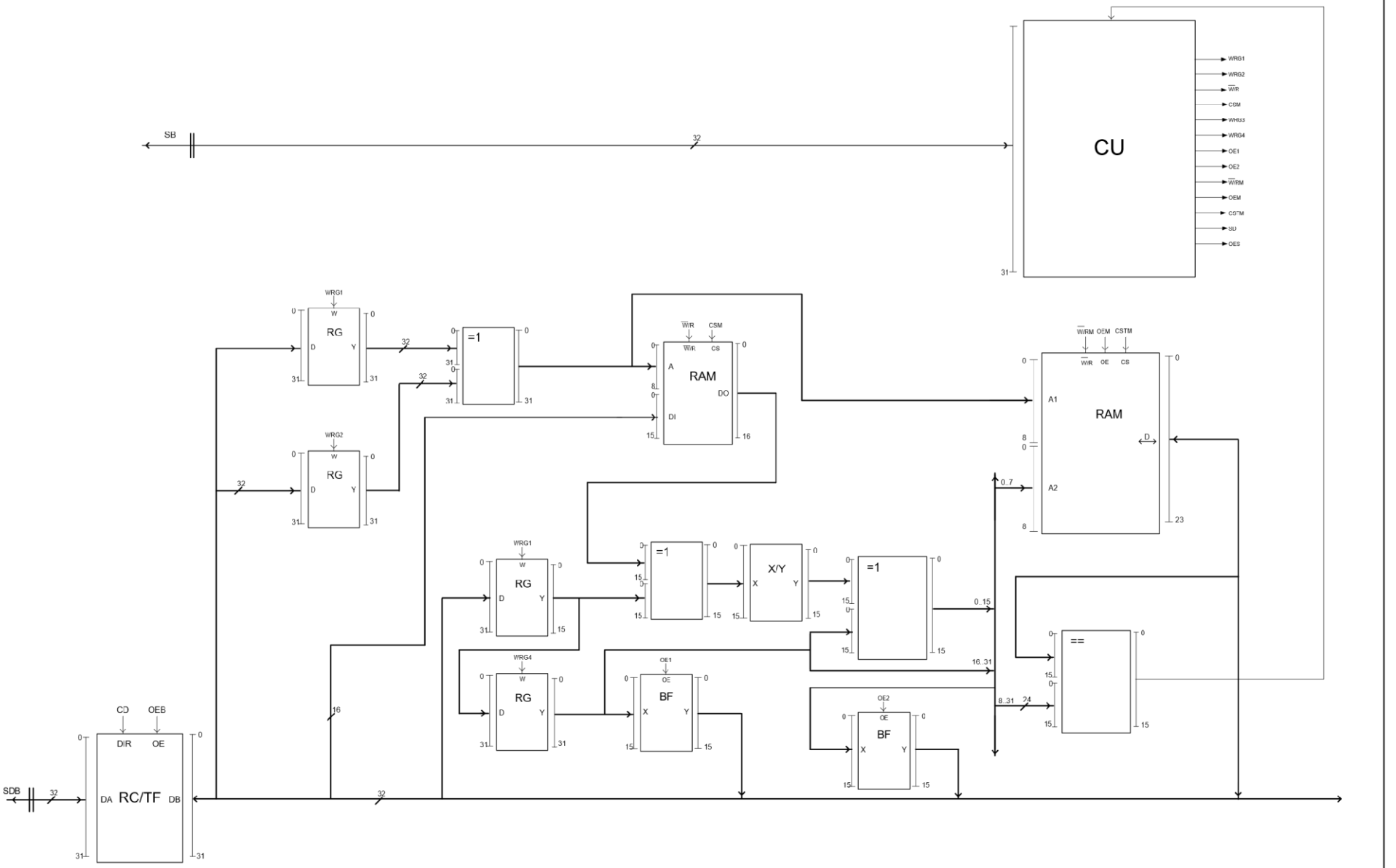
ДОДАТОК Б

Метод та програмні засоби прискореного експоненціювання на полях Галуа

Функціональна схема

Аркушів 1

Київ – 2020 року



Зм.	Лист	№докум.	Підпис	Дата	ДП 4682. 05.000 Д2	
Розроб.		Кот О.С.			Обчислювач експоненти на полях Галуа	Літ. Маса Масштаб
Пров.		Мірошничий			Схема електрична функціональна	Д
Т.контр.						Лист 1 Листів 1
Н.контр.		Симоненко В.П.				НТУУ "КПІ" ФОТ, ІО-61
Зав.		Отрошко С.Г.				

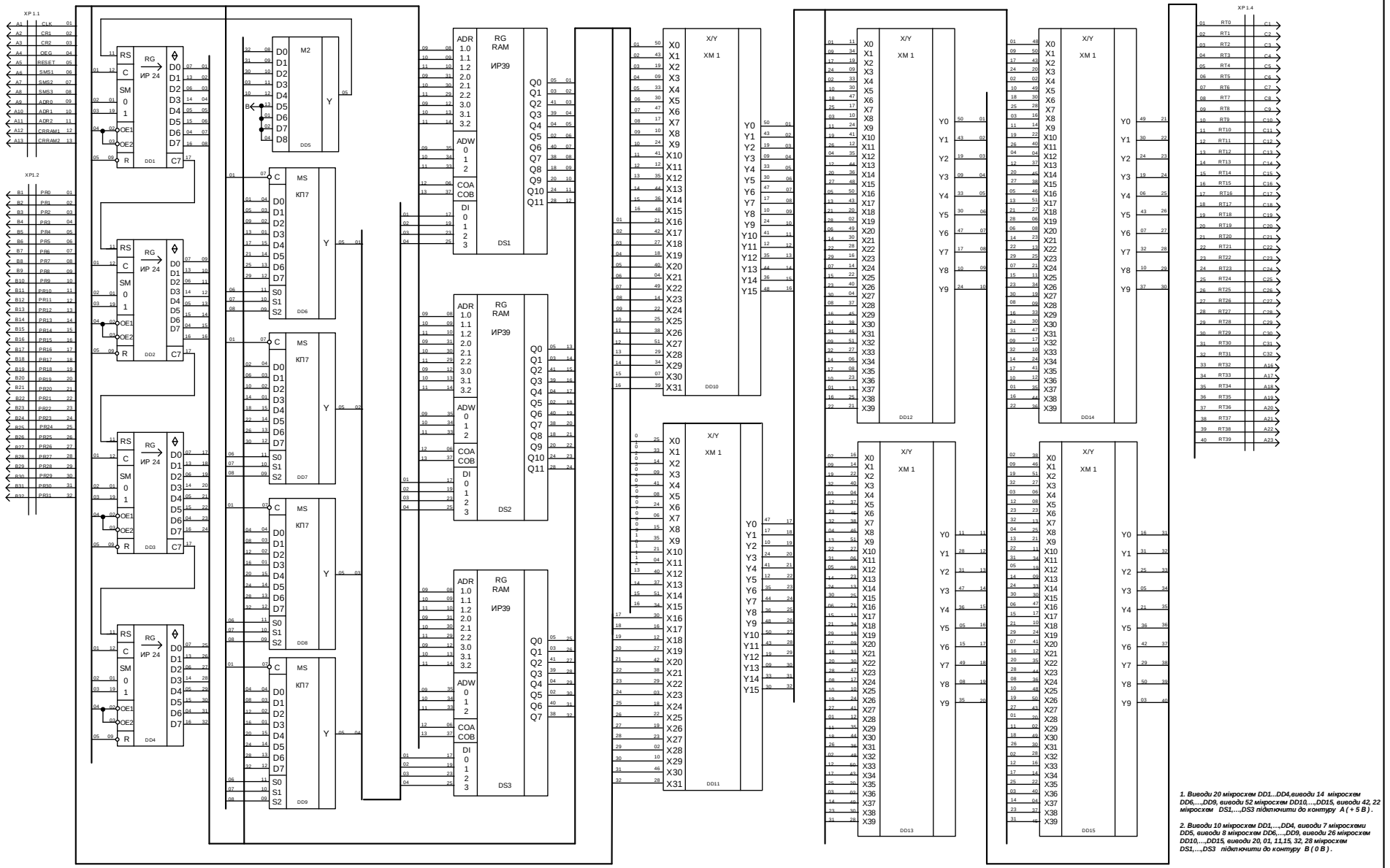
ДОДАТОК В

Метод та програмні засоби прискореного експоненціювання на полях Галуа

Принципова схема

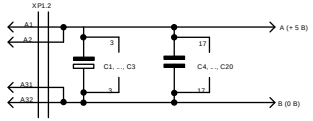
Аркушів 1

Київ – 2020 року



1. Виводи 20 мікросхем DD1..DD4,виводи 14 мікросхем DD6,...,DD9, виводи 52 мікросхем DD10,...,DD15, виводи 42, 22 мікросхем DS1,...,DS3 підключити до контуру А (+ 5 В).

2. Виводи 10 мікросхем DD1,...,DD4, виводи 7 мікросхеми DD5, виводи 8 мікросхем DD6,...,DD9, виводи 26 мікросхем DD10,...,DD15, виводи 20, 01, 11,15, 32, 28 мікросхем DS1,...,DS3 підключити до контуру В (0 В).



						ДП 4682. 06.000 ДЗ	
						Обчислювач експоненти на полі Галуа	Літер. / Маса / Масшт.
Изм.	Лист	N докум.	Підпис	Дата			
Розроб.		Кот О.С.				Схема електрична принципова	А
Перев.		Марюсевський					Аркуш 1 / Аркушів
Т.контр.							
Н.контр.		Симоненко В.					НТУУ "КПІ", гр.ІО-61
Затв.		Симоненко В.					

ДОДАТОК Г

Метод та програмні засоби прискореного експоненціювання на полях Галуа

Лістінг програми прискореного експоненціювання на полях Галуа з застосуванням редукції Монтгомері.

Аркушів 5

Київ – 2020 року

```
from copy import copy
def to_poly(n):
 bitrep = bin(n)[2:]
 return [int(c) for c in bitrep]


def pad_poly(poly, arr_len):
 return [0]*(arr_len-len(poly))+poly

def gadd(a, b):
 return [0 if a[i] == b[i] else 1 for i in range(len(a))]

def gdiv2(n):
 return [0]+n[:-1]

def ton(n_arr):
 return int(''.join(str(i) for i in n_arr), 2)

def shl(n_arr, pos):
 return n_arr[pos:] + [0]*pos


def montgomery_mul(na, nb, np, print_progress=True):
 a = to_poly(na)
 b = to_poly(nb)
 p = to_poly(np)
 ndig = len(p)
 a = pad_poly(a, ndig)
 b = pad_poly(b, ndig)


 y = pad_poly([], ndig)

 if print_progress:
       print("starting calculation...")
 for j in range(ndig-1, 0, -1):
       if b[j] == 1:
             y = gadd(y, a)
       if y[-1] == 1:
             y = gadd(y, p)

       y = gdiv2(y)
       if print_progress:
             if j % (ndig-1)//10 == 0:
                   print("%",end='')

 ny = ton(y)
 if print_progress:
       print()


       print("A*B*R^(-1) rem M = {} ({} binary digits)".format(ny,
len(bin(ny))-2))
       print("")
 return ny

def montgomery_mul_desc(na, nb, np):
 a = to_poly(na)
 b = to_poly(nb)
 p = to_poly(np)
```

```
        ndig = len(p)
        a = pad_poly(a, ndig)
        b = pad_poly(b, ndig)


        y = pad_poly([], ndig)


        print("starting calculation...")
        for j in range(ndig-1, 0, -1):
              print("step {}:".format(ndig-1-j), end='')
              if b[j] == 1:
                    y = gadd(y, a)
                    print("  b[{}] == 1, adding a".format(ndig-1-j))
                    print("    y = {} or {}".format(y, ton(y)))
              if y[-1] == 1:
                    y = gadd(y, p)
                    print("  y[0] == 1, adding p".format(ndig-1-j))
                    print("    y = {} or {}".format(y, ton(y)))

              y = gdiv2(y)
              print("  dividing by 2")
              print("    y = {} or {}".format(y, ton(y)))


        ny = ton(y)
        print("A*B*R^(-1) rem M = {} ({} binary digits)".format(ny,
len(bin(ny))-2))
        print("")
        return ny

      def max_dig(n):
        i = 0
        while n[i] == 0:
              i+=1
        return i

      def g_bigger(n1, n2):
        i = 0
        try:
              while n1[i] == n2[i]:
                    i+=1
        except:
              return False
        if n1[i] == 1:
              return True
        return False



      def calc_c(np):
        p = to_poly(np)
        ndig = len(p)
        p = pad_poly(p, (ndig-1)*2+1)


        r2 = pad_poly([0], (ndig-1)*2+1)
        r2[0] = 1


        while g_bigger(r2, p):
```

```
        r2 = gadd(r2, shl(p, max_dig(p) - max_dig(r2)))


    return(ton(r2))


def montgomery_exp_desc(a, ne, p):
    ndig = len(to_poly(p))

    e = to_poly(ne)

    print("initial values:")
    q = p - 2**(ndig-1)
    print("  q =", q)

    c = calc_c(p)
    print("  c =", c)

    b = montgomery_mul(a, c, p, print_progress=False)


    y = q
    for j in range(0, len(e)):
        print("step:",len(e)-j-1)
        y = montgomery_mul(y, y, p, print_progress=False)
        print("  after squaring:\n    y =", y)
        if e[j] == 1:
            oldy = y
            y = montgomery_mul(y, b, p, print_progress=False)
            print("  e[j] == 1, y =", oldy, "*", b, "\n    y =", y)

    y = montgomery_mul(y, 1, p, print_progress=False)
    print("last step:\n  y = y*1\n    y =",y)
    return

def montgomery_mul(na, nb, np, print_progress=True):
    a = to_poly(na)
    b = to_poly(nb)
    p = to_poly(np)
    ndig = len(p)
    a = pad_poly(a, ndig)
    b = pad_poly(b, ndig)


    y = pad_poly([], ndig)

    if print_progress:
        print("starting calculation...")
    for j in range(ndig-1, 0, -1):
        if b[j] == 1:
            y = gadd(y, a)
        if y[-1] == 1:
            y = gadd(y, p)

        y = gdiv2(y)
        if print_progress:
            if j % (ndig-1)//10 == 0:
                print("%",end='')

    ny = ton(y)
    if print_progress:
        print()
```

```
            print("A*B*R^(-1) rem M = {} ({} binary digits)".format(ny,
len(bin(ny))-2))
            print("")
      return ny

     def montgomery_sqr_desc(na, np):
      a = to_poly(na)
      b = to_poly(nb)
      p = to_poly(np)
      ndig = len(p)
      a = pad_poly(a, ndig)
      b = pad_poly(b, ndig)


      y = pad_poly([], ndig)


      print("starting calculation...")
      for j in range(ndig-1, 0, -1):
            print("step {}:".format(ndig-1-j), end='')
            if b[j] == 1:
                  y = gadd(y, a)
                  print("  b[{}] == 1, adding a".format(ndig-1-j))
                  print("    y = {} or {}".format(y, ton(y)))
            if y[-1] == 1:
                  y = gadd(y, p)
                  print("  y[0] == 1, adding p".format(ndig-1-j))
                  print("    y = {} or {}".format(y, ton(y)))

            y = gdiv2(y)
            print("  dividing by 2")
            print("    y = {} or {}".format(y, ton(y)))


      ny = ton(y)
      print("A*B*R^(-1) rem M = {} ({} binary digits)".format(ny,
len(bin(ny))-2))
      print("")
      return ny


     def montgomery_exp(a, ne, p):
      ndig = len(to_poly(p))

      e = to_poly(ne)


      q = p - 2**(ndig-1)

      c = calc_c(p)

      b = montgomery_mul(a, c, p, print_progress=False)


      y = q
      for j in range(0, len(e)):
            y = montgomery_mul(y, y, p, print_progress=False)
            if e[j] == 1:
                  oldy = y
                  y = montgomery_mul(y, b, p, print_progress=False)
```

```
      y = montgomery_mul(y, 1, p, print_progress=False)
      return y

    def console():
      cmd = ''
      while cmd != 'quit':
            cmd = input("cmd:>")
            if cmd == 'mul_desc':
                  a = int(input('a:'))
                  b = int(input('b:'))
                  p = int(input('p:'))
                  montgomery_mul_desc(a, b, p)
            elif cmd == 'mul':
                  a = int(input('a:'))
                  b = int(input('b:'))
                  p = int(input('p:'))
                  montgomery_mul(a, b, p)
            elif cmd == 'exp_desc':
                  a = int(input('a:'))
                  e = int(input('e:'))
                  p = int(input('p:'))
                  montgomery_exp_desc(a, e, p)
            elif cmd == 'exp':
                  a = int(input('a:'))
                  e = int(input('e:'))
                  p = int(input('p:'))
                  print("y =", montgomery_exp(a, e, p))

        mode = input("Enter mode(s or v): ")
        if (mode == "v"):
            s1 = int(input("Enter s1: "))
            s2 = int(input("Enter s2: "))
        value = int(input("Enter value (16 bit): "))

        if mode == "s":
            s1, s2 = dsa_sign(dsa_key.get("Q"), dsa_key.get("P"),
dsa_key.get("G"), dsa_key.get("priv"), value)
            print("S1:" + str(s1))
            print("S2:" + str(s2))
        elif mode == "v":
            result = dsa_verify(s1, s2, dsa_key.get("G"), dsa_key.get("P"),
dsa_key.get("Q"), dsa_key.get("pub"), value)
            print("Result:" + str(result))
        else:
            pass


    console()
```